UNIVERSIDADE DO VALE DO TAQUARI – UNIVATES
CURSO DE ENGENHARIA DA COMPUTAÇÃO

DESENVOLVIMENTO DE UMA FERRAMENTA PARA MIGRAÇÃO DE BANCOS DE DADOS

Vítor Soares Vian

Lajeado, dezembro de 2020.

Vítor Soares Vian

DESENVOLVIMENTO DE UMA FERRAMENTA PARA MIGRAÇÃO DE BANCOS DE DADOS

Monografia apresentada no componente curricular Trabalho de Conclusão de Curso II, do curso de Engenharia da Computação, da Universidade do Vale do Taquari - Univates, como requisito para a obtenção do título de bacharel em Engenharia da Computação.

Orientador: Prof. Me. Willian Valmorbida

Lajeado, dezembro de 2020

AGRADECIMENTOS

Primeiramente agradeço à minha família, que tanto lutou para que eu pudesse estudar e ter um diploma em mãos, obrigado por todo apoio, confiança e compreensão. Um agradecimento especial à minha mãe Mirian, que sempre esteve comigo, me apoiando em todos os momentos nessa jornada acadêmica. Sem seu apoio não seria possível a realização desse sonho!

Agradeço ao meu orientador, professor Willian Valmorbida, pelos sábios conselhos, orientações e correções, por todo companheirismo e dedicação ao longo do desenvolvimento do trabalho.

Também deixo meu agradecimento especial a todos professores, coordenadores, funcionários técnico-administrativos, colegas e demais funcionários da instituição de ensino Univates, todos tiveram um papel importante em minha jornada acadêmica e, por que não dizer, na vida pessoal.

RESUMO

Com uma necessidade constante de atualização e o uso de tecnologias que atendam melhor as demandas de determinado ramo profissional, não é incomum a troca de sistemas de informações por outros e, para isso, é necessário migrar os dados do sistema antigo para o novo, processo conhecido como migração de dados, que é uma das motivações da ferramenta proposta neste trabalho, visto a complexidade e tempo gasto nesse processo. Para isso, a fundamentação teórica elenca uma bibliografia que visa esclarecer os conceitos envolvidos nesse processo que foi tratada utilizando-se como metodologia a pesquisa bibliográfica e experimental. O software apresentado neste estudo tem por objetivo abstrair a complexidade da migração de dados, reduzindo o tempo gasto nesse processo, sendo que o usuário fica responsável apenas por configurá-lo através de um *template* de migração, que pode ser reutilizado em processos futuros de mesma origem e destino. Como forma de testar a usabilidade da ferramenta foi aplicado um questionário a desenvolvedores que já trabalham profissionalmente com softwares. Como resultado apresenta-se a forma como foi desenvolvida a ferramenta bem como a comparação do tempo de migração utilizando a ferramenta e um processo sem ela, tendo a ferramenta sido mais rápida e ágil no processo de migração.

Palavras-chave: Banco de dados. Migração de dados. Sistemas de informação.

ABSTRACT

With the constant necessity of update and the use of technologies to better assist the demands from a certain professional branch, it is not unusual the trade of information systems for others and, to do so, it is necessary to migrate the data from the old system to the new one, which is known as data migration that is the one of the motivations of the tool proposed in this work. In this regard, the theoretical basis lists a bibliography that aims to enlighten the concepts involved in this process, which was treated using the bibliographical and experimental research as methodology. The software referred in this research has as main objective to abstract the complexity in migrating the data by reducing the time spent in this process and to make the user responsible only for configuring it through a migration template, which can be reused in future similar processes. To test the tool usability was carried out a questionnaire with individuals that already work with software development. As results, we present the form with which we develop the tool and a process without it; In this cases, the process that used the tool proved itself to faster and more agile in migrating data.

Keywords: Database. Data migration. Information systems.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS

API	Application Programming Interface
BD	Banco de dados
CSV	Comma-Separated Values
DBA	Database Administrator
DDL	Data Definition Language
DML	Data Manipulation Language
DW	Data warehouse
ETL	Extract, Transform, Load
ER	Entidade-Relacionamento
ERP	Enterprise Resource Planning
EVTE	Estudo de viabilidade técnica e econômica
IDE	Integrated Development Environment
JVM	Java Virtual Machine
JSON	JavaScript Object Notation
MR	Modelo Relacional
MVP	Minimum Viable Product
SGBD	Sistemas de Gerenciamento de Banco de Dados
SI	Sistema de Informação
SQL	Structured Query Language
TXT	Text File
ZIP	Arquivo compactado no formato zip

SUMÁRIO

1 INTRODUÇÃO

Com o passar dos anos, diversas empresas vêm aprimorando a forma como gerenciam, guardam e manipulam dados, de maneira que esse bem tão valioso para as organizações merece uma atenção especial, pois, munindo-se de dados, é possível gerar informações que, por sua vez, geram conhecimento que pode conferir vantagens competitivas a uma organização, bem como melhorias em processos internos, aumento de vendas, entre outros.

Nas organizações, uma das formas mais eficientes de manipular os dados é utilizando sistemas ou programas de computadores (softwares) que, para João (2012), podem ser definidos como um conjunto de componentes que se relacionam de forma a recuperar, armazenar e distribuir informações que podem ajudar na tomada de decisão, de coordenação ou de controle de uma organização.

Nesse sentido, grande parte dos dados de empresas estão armazenados em sistemas que utilizam banco de dados que, segundo Elmasri e Navathe (2018) é definido como a relação entre diversos dados que têm valores implícitos, tais como nomes, números de telefones, endereços, dentre outros.

Por isso, um banco de dados é uma das formas mais utilizadas por programas de computadores para armazenar as informações, porém, quando precisamos trocar de sistema (para atender regras de negócios, atualização de versões etc.) precisamos passar por um processo conhecido como migração de dados que, conforme ensina Mendonça (2009, p. 22) “é uma etapa fundamental do processo de troca de sistemas, levando em consideração a importância que os dados têm para uma organização, onde a perda dos dados pode ocasionar prejuízos financeiros”.

Ainda sob a perspectiva de Mendonça (2009), quando os dados de que um novo sistema necessita provém de alguma outra fonte, eles devem ser migrados e

não criados novamente no novo sistema. Esse processo de migrar os dados de um sistema para outro é chamado pelo nome de migração de dados.

Levando em consideração esse panorama inicial, o presente trabalho propõe-se a desenvolver uma aplicação web para auxiliar no processo de migração de dados entre sistemas, visando torná-lo mais ágeis, mantendo a integridade das informações e garantindo a segurança necessária nesse processo.

1.1 Justificativa

Com um mundo cada vez mais conectado e com a necessidade de informações é quase impossível uma empresa sobreviver sem sistemas de informação, uma vez que eles estão presentes no cotidiano delas, guardando, gerenciando e disponibilizando dados para que se possa gerar informações. Além disso, a troca dos sistemas internos da empresa também é uma ação comum, que busca sempre softwares que atendam melhor e de maneira mais eficaz suas necessidades.

Para isso, faz-se necessária a migração dos dados de sistemas antigos para sistemas novos. Porém, essa tarefa árdua pode ser complicada e levar muito tempo ao longo de toda sua realização, por isso, um sistema especializado em migração de dados se faz necessário para garantir estabilidade, confiabilidade e integridade a esse processo.

Existem diversos motivos para a utilização de uma ferramenta de automatização de migração, como segurança, eficiência e facilidade. Nesse sentido, as ferramentas estudadas neste trabalho buscam, em sua totalidade, garantir uma forma fácil e automatizada de o usuário migrar seus dados com segurança.

1.2 Definição do problema

Conforme a quantidade de dados de uma empresa cresce, a forma como ela lida com eles vai se tornando mais complexa, havendo a necessidade de pensar em alternativas para se relacionar com esses dados, como armazená-los segmentá-los em múltiplos bancos de dados e também recuperar essa informação posteriormente. Uma eventual migração de dados também terá sua complexidade aumentada com o crescimento desses aspectos do gerenciamento de dados, logo, a criação de scripts manuais e migrações utilizando linguagem nativa do banco de dados – como a

Structured Query Language (SQL), por exemplo – ficam inviáveis ou extremamente complexas de serem realizadas.

Unindo esses problemas a outros que podem acontecer, tais como o grande tempo investido em migração de dados e a falta de experiência em migrações, pode-se ter o acarretamento de perdas de dados ao final do processo. Para tanto, a fim de solucionar os problemas, pode-se fazer uso de uma ferramenta específica para o processo de migração, garantindo que ele seja seguro e transparente para o usuário. Com base nisso alguns problemas que podemos destacar são:

- É possível diminuir o tempo de migração e torná-lo mais ágil através de uma ferramenta especializada?
- Como migrar dados sem ter pessoas especialistas nesse processo?

Com base nos nesses questionamentos, ainda se levanta a seguinte problemática: uma ferramenta especializada no processo de migração de dados pode auxiliar de maneira a reduzir o tempo dessa atividade?

1.3 Objetivo geral

Automatizar, simplificar, e diminuir o tempo gasto com o processo de migração de dados entre sistemas, através do desenvolvimento de um *software* web.

1.4 Objetivos específicos

Para alcançar o objetivo geral, foram definidos os objetivos específicos abaixo:

- Entender o funcionamento e a diferença entre os Sistemas de Gerenciamento de Banco de Dados (SGBD) propostos no presente trabalho;
- Propor, gerenciar e estudar uma arquitetura de containers para viabilizar o processo de migração entre diferentes bancos de dados;
- Implementar um protótipo da ferramenta proposta;
- Planejar e executar um plano de testes utilizando o protótipo desenvolvido;
- Confrontar os resultados obtidos pelo uso da ferramenta, comparando um processo manual (utilizando scripts SQL) com o uso da ferramenta.
- Desenvolvimento de um *software* web de migração de dados.

1.4 Delimitação do tema

O presente estudo compreende o desenvolvimento de uma ferramenta (sistema web) para auxiliar o processo de migração de software, dando suporte para uso em dois SGBD específicos, o PostgreSQL e o MySQL. Esse sistema web será responsável apenas pela migração dos dados, não fazendo qualquer tipo de validação deles e sem alterações no *schema* dos SGBD. Além disso, ele será desenvolvido com foco em pequenas e médias bases de dados, estando limitado pelos navegadores web que não aceitam arquivos muito pesados.

1.5 Estrutura do trabalho

Este trabalho está dividido em oito capítulos, sendo que o primeiro capítulo apresenta uma introdução do assunto abordado. Já no segundo apresenta-se o embasamento teórico, enquanto no terceiro estão os trabalhos relacionados ao programa que se almeja desenvolver. No quarto, por sua vez, é apresentada a metodologia, seguida da especificação do projeto, no quinto capítulo. No sexto capítulo é apresentado a implementação da ferramenta e os resultados e as considerações finais compõem o sétimo e oitavo capítulos respectivamente.

2 REFERENCIAL TEÓRICO

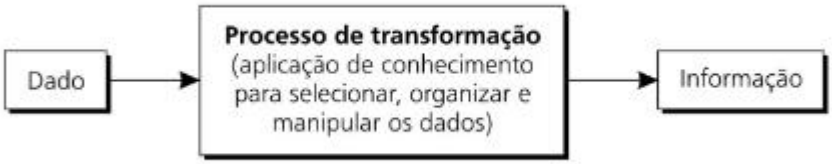
Neste capítulo, descreve-se o referencial teórico que, conforme Gil (2017), pode ser entendido como um estudo que visa explorar o tema pretendido e que tem por objetivo proporcionar ao autor uma base de conhecimento sobre o tema proposto, o que é de suma importância para a definição de um problema de pesquisa claro e objetivo.

2.1 Dado *versus* informação

Dado, segundo Setzer (2015), pode ser definido como um conjunto de símbolos com significado que podem ser quantificados ou que são quantificáveis, portanto, como o próprio autor determina, um arquivo em texto, fotos, animações, dentre outros, podem ser considerados dados. Ainda segundo Setzer (2015) os dados, por serem quantificáveis, podem ser armazenados e processados por um computador, pois para ele todo dado é obrigatoriamente uma entidade matemática. Já conforme Audy, Andrade e Cidral (2007), um dado consiste em um fato primitivo tal como, o número de um telefone, o nome de um objeto, ou suas formas representativas: imagens, sons etc., que por si só podem fazer sentido ou não a um determinado processo em particular.

Com um conjunto de dados organizado e estruturado, podemos gerar informações, pois, como dizem Audy, Andrade e Cidral (2007), a informação são dados concatenados e transformados, cujo conteúdo serve para algum fim específico, gerando um valor em cima dos fatos. A Figura 1, a seguir, exemplifica a relação entre dado e informação.

Figura 1 - Relação entre dado e informação



Fonte: Audy, Andrade e Cidral (2007, p. 89)

Audy, Andrade e Cidral (2007) elucidam também que a informação já não é mais um diferencial entre as empresas, passando a ser um elemento essencial da sobrevivência corporativa. Pensando nisso, gerir as informações de uma maneira correta e precisa se faz necessário, como com o uso de sistemas de informação eficazes.

2.2 Sistemas de informação

Conforme Batista (2012), a maioria dos gestores de negócios precisam tomar decisões com base em fatos (dados) e informações do dia a dia, porém a quantidade de dados gerados diariamente inviabiliza seu estudo detalhado. Por isso, torna-se necessário o uso de ferramentas para agrupar, filtrar e organizar os dados conforme a relevância desses dados.

Nesse sentido, os sistemas de informação (SI) podem ser úteis, pois como explana João (2012), um SI pode ser definido como um aglomerado de componentes que se relacionam entre si e coletam, processam, armazenam e distribuem informações utilizadas para a tomada de decisões e organização de uma empresa. João (2012) também elenca alguns objetivos organizacionais (elucidando com exemplos) dos sistemas de informação, tais como: excelência operacional; novos produtos; serviços e modelos de negócios; relacionamentos mais estreitos com clientes e fornecedores; melhor tomada de decisões; vantagens competitivas e sobrevivência. Estes seis objetivos organizacionais são detalhados no Quadro 1, conforme segue.

Quadro 1 - Objetivos organizacionais

Objetivo	Descrição
Excelência operacional	Busca melhorar a eficiência de operações da empresa. Como exemplo, há o Walmart, dos Estados Unidos, que faturou 400 bilhões de dólares em 2008 pois, com seu SI, foi possível conectar as diversas filiais e, após uma venda, o fornecedor era comunicado para repor o produto.
Novos produtos, serviços e modelos de negócios	São gerados a partir das informações armazenadas. Os modelos de negócios servem como diretrizes de como a empresa deve gerir, organizar entregar e vender um produto buscando gerar valor a partir dele.
Relacionamentos mais estreitos com clientes e fornecedores	No relacionamento mais estreito com fornecedores e clientes, temos um laço mais próximo graças ao uso dos SI, que geram uma comunicação mais efetiva e mais rápida.
Melhor tomada de decisões	Melhores tomadas de decisões com base nas informações armazenadas em sistemas da empresa.
Vantagens competitivas	Vantagens competitivas podem ser extraídas de conhecimentos gerados pelo uso dos sistemas de informação.
Sobrevivência	A sobrevivência é outro objetivo, pois, no atual mundo conectado em que vivemos, é muito difícil para uma empresa sobreviver sem uma forma efetiva de armazenar seus dados.

Fonte: Elaborado pelo autor e adaptado de João (2012).

Para alcançar esses objetivos é necessária uma forma de armazenar e de lidar com os dados que a empresa precisa, sendo que, para isso, muitos dos sistemas de informação utilizam bancos de dados para o armazenamento dos respectivos dados.

2.3 Banco de dados

Conforme Amadeu (2015) os bancos de dados (BD) são essenciais em nosso cotidiano e estão em simples tarefas do nosso dia a dia, como depositar ou sacar dinheiro no banco, em bibliotecas virtuais das universidades, em processos de

compras on-line, dentre outros. Um banco de dados pode ser definido, resumidamente, como uma coleção de dados relacionados e armazenados.

Ampliando esse conceito, segundo Elmasri e Navathe (2018) um banco de dados é uma coleção lógica e coerente de dados com algum significado, que atende alguma proposta específica, possuindo um grupo de usuários definidos e algumas aplicações pré-concebidas, conforme a necessidade desses usuários. Em outras palavras, um banco de dados possui algumas fontes distintas cujos dados são advindos de algumas interações com o mundo real e com usuários interessados em seus conteúdos.

Já para Amadeu (2015), um banco de dados precisa ser eficiente, possibilitando ao usuário o acesso de suas informações de forma imediata ou dentro de poucos segundos e, para garantir isso, um banco de dados precisa observar os seguintes fatores: redundância, consistência e integração. A redundância é caracterizada por dados duplicados armazenados, o que gera custo de armazenamento desnecessário, ao passo que a consistência acontece quando um dado sofre uma alteração solicitada, e a integração que, por sua vez, é a capacidade de manter os dados disponíveis para diferentes fontes consumidoras.

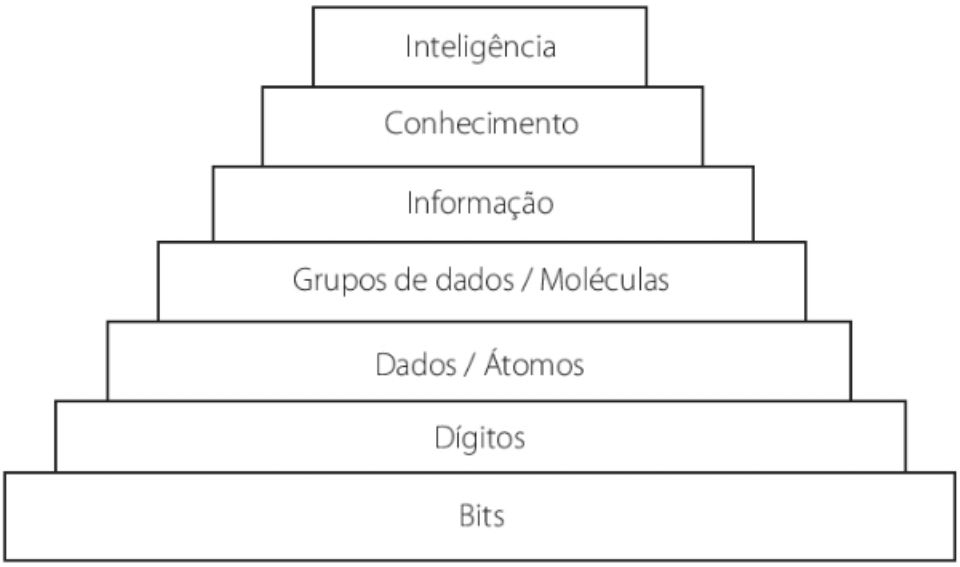
O processo de guardar uma informação em um banco de dados é bem diferente de guardarmos uma lista de nomes no Microsoft Excel, em que há uma série de passos para que a informação seja armazenada. Como exemplifica Amadeu (2015), o processo de armazenamento de informações em um BD segue os seguintes níveis:

- No nível 1, tem-se um arquivo com sequências de 0 e 1 denominados bits;
- No nível 2, tem-se uma cadeia de oito bits (ou byte), que corresponde a um caractere (podendo ser um número ou letra);
- No nível 3, já são formados os dados ou átomos;
- No nível 4, tem-se os grupos de dados, pois dados precisam se complementar em um agrupamento, por exemplo, o nome “Ana” precisa de dados relacionados, como idade, endereço e telefone;
- No nível 5, contextualiza-se aquele grupo de dados: porque armazenar esses dados da Ana? Ela comprou um produto, necessita de um serviço? É nesse momento que é gerada a informação;
- Unindo-se essas informações, no nível 6 gera-se o conhecimento, que é quando o administrador dessas informações as compreende e consegue tirar algum benefício delas;

- No nível 7, por fim, fundamenta-se a inteligência, que é poder agir ou tomar providências com base nas informações armazenadas.

A Figura 2, elencada a seguir, sintetiza esses níveis de maneira hierárquica.

Figura 2 - Níveis de armazenamento de informações



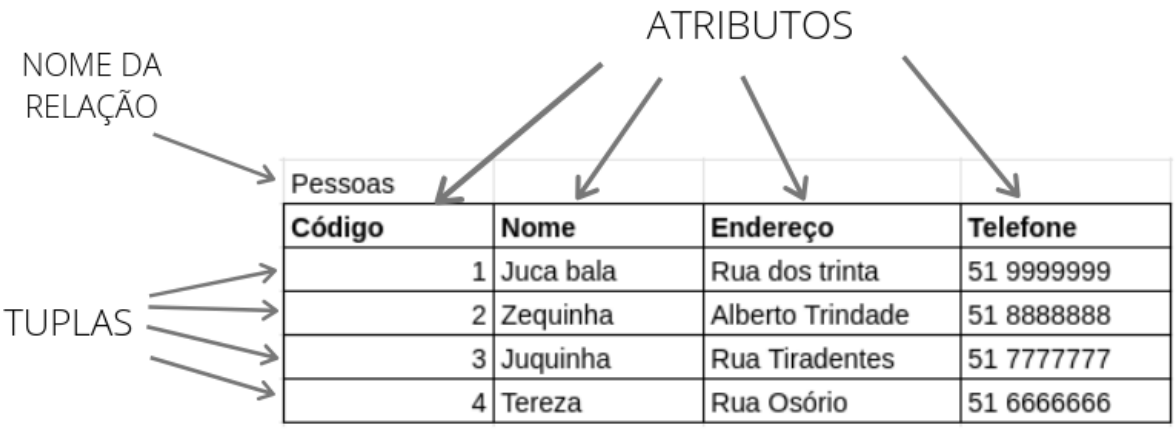
Fonte: Medeiros (2013 p. 21).

Além do mencionado, para acessarmos as informações contidas em um banco de dados, precisamos de uma estrutura robusta e bem organizada. A esse respeito, no próximo capítulo, discorre-se acerca de como os dados são organizados em um BD.

2.3.1 Estrutura dos bancos de dados relacionais

Como exemplificam Elmasri e Navathe (2018), um modelo relacional de banco de dados é representado por relações, que são definidas como algo similar a uma tabela com valores e, se aplicarmos a ele os termos do Modelo Relacional (MR), denominam-se linhas como tuplas, colunas como atributos e a tabela como relação. Para um melhor entendimento, a Figura 3 detalha esses elementos.

Figura 3 - Estrutura de um banco de dados



Pessoas

Código	Nome	Endereço	Telefone
1	Juca bala	Rua dos trinta	51 9999999
2	Zequinha	Alberto Trindade	51 8888888
3	Juquinha	Rua Tiradentes	51 7777777
4	Tereza	Rua Osório	51 6666666

Fonte: Elaborado pelo autor, adaptado de Costa (2011, p. 33).

2.3.2 Relação

Conforme Medeiros (2013) uma relação, uma lista ou uma tabela é um agrupamento de dados com um mesmo contexto, de modo que a relação contém um identificador próprio (em um banco de dados podem haver diversas relações). Já para Costa (2011) um banco de dados é um conjunto de relações, considerando que uma relação é similar a uma tabela de valores por meio da aplicação da terminologia do modelo relacional.

2.3.3 Tuplas

Para Amadeu (2015) uma tupla pode ser definida como uma linha do banco de dados. Assim, um registro ou tupla é um conjunto de campos com valores atribuídos em uma determinada relação, sendo também a unidade básica de inserção e de recuperação de informações em um banco de dados. Elas podem ser chamadas de linhas, pois no banco de dados cada registro é representado por uma linha. Logo, se um banco de clientes possui 20.000 linhas, ele possui 20.000 registros de clientes (ALVES, 2014).

2.3.4 Atributos

Para Medeiros (2013) um atributo de entidade representa as características dela, podendo conter uma faixa de valores ou de domínio que descreve o tipo de dado que é apresentado. Além disso, eles podem ser atômicos (simples) ou não atômicos (compostos). Podemos também ter atributos identificadores ou não, sendo que os identificadores servem para identificar a tupla na entidade. Na Figura 3, por exemplo, o campo Código tem essa função. Já para Amadeu (2015) um atributo da entidade pode ser definido como um cabeçalho da coluna.

2.4 Sistema Gerenciador de Banco de Dados

Para Elmasri e Navathe (2018) um Sistema Gerenciador de Banco de Dados (SGBD) é um conjunto de softwares que permite que o usuário ou que o sistema interaja com um banco de dados propriamente dito. Portanto, um SGBD é um software que tem o propósito de facilitar processos de construção, de consumação, de manipulação e de compartilhamento dos dados contidos em um BD entre os diferentes usuário e aplicações. Um SGBD tem, principalmente, as funções de construção, de manipulação, de compartilhamento e de proteção de dados:

- A construção de dados consiste em armazenar os dados de um DB em uma mídia de armazenamento controlada pelo SGBD;
- A manipulação dos dados consiste em duas operações principais, sendo a primeira a recuperação dos dados para consulta e a segunda, a atualização dos registros para refletir as mudanças;
- O compartilhamento consiste em uma partilha correta da informação entre usuários e aplicações que consultem e consumam os dados presentes no BD;
- A proteção dos dados consiste nos processos feitos pelo SGBD para garantir a proteção contra o mau funcionamento do BD tanto em questões de hardware e de software, e também no controle de acesso aos dados por usuários.

Date (2004), descreve o SGBD como um software que fica entre o banco de dados físico e o usuário, sendo que todas as requisições feitas ao BD são tratadas por ele, que fica, assim, responsável por isolar dos usuários do banco de dados certos detalhes dos tratamentos feitos a nível de hardware. Logo, o SGBD fornece aos seus usuários uma visão elevada do banco de dados e, para que isso seja possível, a

maioria dos SGBD admite a linguagem SQL como sendo uma forma de comunicação entre o usuário e o banco de dados.

2.5 Structured Query Language

A *Structured Query Language* (SQL) ou Linguagem de Consulta Estruturada, em tradução livre, foi criada pela IBM Research, no início da década de 1970, para o protótipo de um sistema de banco de dados chamado System R (DATE, 2004). Segundo Elmasri e Navathe (2018), a linguagem SQL é baseada em álgebra e cálculo relacional, o que gerou facilidade para o usuário, se considerarmos as linguagens nativas dos SGBD.

Já conforme Silberschatz, Korth e Sudarshan (1999), apesar de ser conhecida como uma linguagem de consulta, a SQL permite incluir, atualizar, deletar e configurar recursos de um banco de dados. Nesse sentido, Elmasri e Navathe (2018) afirmam que a SQL pode ser dividida em dois tipos, conforme seus objetivos: *Data Definition Language* (DDL) e *Data Manipulation Language* (DML). A DDL refere-se aos comandos utilizados pelo *Database Administrator* (DBA) e pelos projetistas de dados para definir a estrutura do banco de dados. Já a DML é um conjunto de comandos que tem por objetivo incluir, alterar e deletar os dados armazenados.

2.6 Migração de dados

Elmasri e Navathe (2018) ensinam que algumas organizações têm delegado a administração e o armazenamento de suas informações a grandes e complexos sistemas e SGBDs, o que torna difícil a substituição desses dados, de modo que a organização se torna atrelada ao conjunto de tecnologias escolhidas. Porém, muitas vezes, o custo pode ser alto se um sistema grande e complexo não puder evoluir e então torna-se necessária a migração de dados.

Para Mendonça (2009), quando os dados de que um novo sistema necessita vem de outros sistemas e sua quantidade é considerável, esses dados devem ser migrados e não recriados no novo sistema. Esse processo de passar dados existentes de uma aplicação para outra é chamado de migração de dados. Conforme Mello (2009) há vários fatores para se levar em conta em um processo de migração, sendo os dois principais a análise da motivação da migração e o fato de se a tecnologia escolhida é a melhor para atender as novas necessidades.

Ainda, para Mello (2009), os riscos de uma migração de dados são bem elevados. Portanto, deve-se fazer uma análise completa tanto da tecnologia antiga quanto da nova para entender se os riscos são favoráveis.

2.7 Processo de migração de dados

Conforme explana Mendonça (2009) o processo de migração de dados é complexo e, por essa razão, deve-se avaliar e migrar a menor quantia de dados necessários para que a aplicação-destino esteja funcional, porém nem sempre é possível uma migração mínima, sendo necessário migrar todos os dados, cenário que requer um maior planejamento e esforço. Por isso, normalmente o processo de migração é separado em extração e carga de dados.

A etapa de extração refere-se a extrair os dados da base existente e armazená-los em um arquivo. Se o volume de dados for pequeno, os dados podem ser colocados em arquivos-texto, porém em volumes maiores procura-se uma mídia de armazenamento. Já na etapa de carga de dados, eles são transferidos para a base de destino, podendo ou não passar por um processo chamado transformação de dados, cuja principal finalidade é transformar e preparar os dados para serem inseridos na base de destino.

2.8 Estratégias de migração de dados

Uma migração de dados permite que os dados sejam modificados para assim serem compatíveis com o banco de dados de destino, porém, deve-se ter muita cautela e competência nesse processo, uma vez que é importante que a migração seja feita com qualidade e integridade, preservando os dados e garantindo que nenhum deles seja perdido. Para isso, existem duas estratégias conhecidas: o *Big Bang* e o *Trickle* (ROMAN; NOTARI, 2018).

As migrações que adotam a estratégia do *Big Bang*, realizam todo o processo de uma vez só, devendo o sistema a ser migrado ficar fora do ar durante o processo de extração e carga de dados. Para esse tipo de migração, deve-se adotar a estratégia de migrar o sistema no menor tempo possível, o que acaba acarretando em uma pressão para a finalização do processo e de seus testes, por isso esse processo normalmente ocorre em finais de semana ou feriados (MENDONÇA, 2009).

Em migrações que adotam a estratégia *Trickle*, os dados são migrados de forma incremental, ou seja, os dois sistemas executam de forma paralela e os dados são migrados em partes. Para tal, deve-se ter um processo em tempo real que atualiza o dado nos dois sistemas, a fim de que ambos estejam atualizados, porém, essa estratégia adiciona complexidade ao processo de migração.

Assim, há duas formas de implementar essa estratégia: na primeira, o usuário deve utilizar os dois sistemas, buscando informações em cada um, dependendo daquela que ele necessita no momento, ao passo que na segunda os usuários continuam usando o sistema antigo até que a migração termine, porém, qualquer dado atualizado ou inserido no sistema antigo deve ser replicado para o novo, mantendo-o atualizado (MENDONÇA, 2009). O Quadro 2 apresenta um comparativo entre as principais estratégias de migração de dados.

Quadro 2 - Principais estratégias de migração

Características	**Big Bang**	**Trickle**
Necessidade de sincronismo	Não	Sim
Complexidade	Média	Alta
Sistema legado fora do ar	Sim	Não
Dados migram primeiro	Sim	Não
Porte de migração	Pequeno	Média/grande
Interoperabilidade entre as aplicações	Não	Sim
Compatibilidade com as metodologias de migração de sistemas legados	*Butterfly*	*Chicken Little*

Fonte: Mendonça (2009, p. 9)

2.9 Metodologias de migração de dados

Para executar um processo de migração de dados é preciso definir uma metodologia de migração. Neste trabalho, apresentam-se três metodologias de migração, sendo elas: *Extract, Transform and Load* (ETL), *Chicken Little* e *Butterfly*.

2.9.1 *Extract, Transform and Load* (ETL)

Conforme Oliveira e Marcelino (2012), na metodologia ETL há três passos, sendo eles a extração, a transformação e a carga, dos quais o único opcional é a transformação dos dados, que é aplicada apenas em casos de necessidade. Pode-se, então, identificar que essa metodologia é adequada quando temos um ambiente homogêneo, a fim de que esse processo seja uma transposição de dados de um sistema para outro com tratamentos simples antes da carga final.

No processo de extração, podemos ter os dados advindos de diversas fontes, como de outro SGBD, de arquivos, de planilhas, dentre outros. Todavia, isso não importa, porque qualquer dado pode ser ajustado e transformado por programadores, de modo que no processo de extração o mais importante é documentar de forma clara como os dados estão organizados no sistema de origem. Isso para garantir que a próxima etapa (transformação) seja minuciosa e explore todos os recursos necessários para uma migração sem perdas e com as regras de negócios preservadas (OLIVEIRA; MARCELINO, 2012).

O detalhamento e a qualidade da documentação construída no passo anterior vão condizer com o sucesso no desenvolvimento dessa etapa, pois tendo um modelo claro dos dados e das regras de negócio do sistema de origem, o processo de transformação de dados torna-se facilitado. Cabe a essa etapa, então, a definição do mapeamento do conjunto de relacionamentos e dos atributos de origem para os da tabela de destino, assim como os processos de manipulação e de conversões que eles necessitam para se adequar ao novo sistema (OLIVEIRA, MARCELINO, 2012).

A etapa final, por sua vez, consiste em gerar a saída dos dados em comandos da linguagem SQL para, então, serem inseridos no SGBD de destino. Ao final desse processo, sugere-se um processo de validação dos dados migrados através de rotinas ou de um processo manual de validação por amostragem de dados (OLIVEIRA; MARCELINO, 2012).

Além disso, o processo de ETL não é usado apenas em migrações de dados, pois, como defendem Thomsen e Pedersen (2009), o processo de ETL é crucial para projetos de *Data Warehouse* (DW), pois possibilita a extração de dados de diferentes fontes para, posteriormente, serem tratados e finalmente inseridos nos DW. Em um projeto de DW, estima-se que 80% do tempo é gasto em ETL, com isso, pode-se entender a dificuldade e a complexidade desse processo (THOMSEN; PEDERSEN, 2009).

2.9.2 Chicken Little

Essa metodologia é conhecida por ser incremental com objetivo de migrar a aplicação por completo, sendo composta por onze passos que são realizados de forma interativa, pois a migração real é feita em pequenas partes (a partir do princípio *Divide and Conquer*) (OLIVEIRA; MARCELINO, 2012). Os passos são os seguintes:

1. Analisar o sistema legado;
2. Decompor a estrutura do sistema legado;
3. Projetar a interface destino;
4. Projetar a aplicação destino;
5. Projetar a base de dados destino;
6. Instalar o ambiente destino;
7. Criar e instalar os *gateways* necessários;
8. Migrar as bases legadas;
9. Migrar as aplicações legadas;
10. Migrar as interfaces legadas;
11. Mudar para o sistema destino.

Essas onze etapas utilizam *gateways*, que podem ser definidos como módulos de software postos entre componentes de softwares, com o objetivo de mediá-los. Para isso a equipe de migração deve escolher seus métodos de migração, dentre os quais citam-se: o método *Forward* ou *Database First* e o método *Reverse* ou *Database Last* (MENDONÇA, 2009).

O método *Forward* ou *Database First* é caracterizado por migrar o banco de dados primeiramente, enquanto migra a aplicação e as interfaces, sendo que adicionalmente é disponibilizado um *Forward Gateway* para que a aplicação original acesse os dados na base de destino (MENDONÇA, 2009). Segundo Oliveira e

Marcelino (2012), no *Forward* ou *Database First*, os dados são direcionados do sistema legado para o novo sistema e o *gateway* faz com que que o local de origem envie os dados ao de destino.

Já no método *Reverse* ou *Database Last,* a aplicação e as interfaces são migradas primeiro, enquanto os dados permanecem no banco de dados original. Dessa forma, a migração do banco de dados é a última etapa, sendo que é disponibilizado um *Reverse Gateway* para que a aplicação destino acesse os dados na aplicação de origem (MENDONÇA ,2009). Em conformidade com essa perspectiva, Oliveira e Marcelino (2012) dizem que o *Reverse* ou *Database Last* é o contrário do método *Forward*, pois no *Reverse* o sistema de destino busca os dados no sistema de origem, mediado por um *gateway*. Entretanto, nos dois processos ambos os sistemas operam paralelamente, o que adiciona complexidade ao processo de migração, pois manter os dados consistentes e atualizados em dois sistemas heterogêneos é complexo (MENDONÇA, 2009).

2.9.3 Butterfly

Na metodologia *Butterfly*, o sistema legado permanece em execução enquanto, em um ambiente de teste, o novo sistema é desenvolvido e testado antes da migração propriamente dita, de modo que todo esse processo não causa qualquer tipo de transtorno do ponto de vista de migração. No *Butterlfy*, utiliza-se uma engrenagem ou motor de migração entre o sistema de origem e o sistema de destino que é conhecido como *Chrysaliser* (OLIVEIRA; MARCELINO, 2012).

O *Chrysaliser* é um sistema intermediário, cujo propósito é converter os dados de um sistema para outro, uma vez que os dados podem necessitar de ajustes para serem inseridos no sistema de destino. Comumente é necessária a intervenção de um administrador de banco de dados nesse sistema intermediário para incluir as regras de migração (OLIVEIRA; MARCELINO, 2012). Além disso, para Mendonça (2009), essa metodologia leva em consideração o fato de que o sistema legado não estará em produção quando o processo de migração for efetuado. Para isso, há o desenvolvimento de um motor de migração de dados, para que o sistema fique fora do ar o mínimo possível.

3 TRABALHOS RELACIONADOS

Para servir como apoio e forma de estudo dos caminhos já percorridos para migrar dados, foram estudados trabalhos similares ao proposto por este estudo, visando traçar semelhanças, prevenir erros e melhor entender o processo de migração de dados na visão de outros autores.

3.1 Metodologias e estratégias de migração de dados

Oliveira e Marcelino (2012) desenvolveram um trabalho como forma de direção em metodologias para apoiar ferramentas de migração de dados, o qual versa sobre metodologias e estratégias de migração de dados, baseados em bibliografia, o que auxiliou o presente estudo a direcionar e escolher uma metodologia de migração que se adapte à proposta do software que será desenvolvido.

Como metodologias, foram abordadas a *Extract, Transform and Load* (ETL), a *Chicken Little*, a *Cold Turkey* e a *Butterfly*. Conforme o apresentado nos estudos de Oliveira e Marcelino (2012), foi decidido que a metodologia *Butterfly* é a mais adequada para o proposto neste trabalho, pois como os autores exemplificam, nessa metodologia há o uso de um sistema intermediário responsável por transformar os dados, chamado *Crystalizer*.

Apesar do ETL também ser uma metodologia muito próxima ao que se deseja desenvolver como *software* mediador, segundo Oliveira e Marcelino (2012) a etapa de transformação de dados é opcional na metodologia ETL. Contudo, no que se pretende com o atual estudo, a transformação sempre ocorrerá mesmo que não tenhamos alteração no conteúdo do conjunto de dados.

3.2 Desenvolvimento de uma ferramenta para auxiliar na migração de dados entre sistemas ERP

Vogel (2019) descreve o desenvolvimento de uma plataforma intermediadora para auxiliar o processo de migração de dados, tendo como público-alvo pequenas e médias empresas de tecnologia que utilizem processos de migração de dados de sistemas: os *Enterprise Resource Planning* (ERP).

O autor apresenta uma interface de usuário limpa, objetiva e muito bem estruturada, o que reflete em possíveis ideias de como construir uma interface para softwares com objetivos semelhantes. As diferenças entre a plataforma desenvolvida por Vogel (2019) e as propostas deste estudo estão detalhadas a seguir:

- Ele baseia-se em um processo de migração no qual as bases de origem são diferentes, porém a base de destino é conhecida, ou seja, o esquema do banco de dados destino é sempre o mesmo;
- O software é instalado direto na máquina em que será processada a migração, o que acarreta em uma série de dependências, como ter instalado na máquina o *Java Virtual Machine* (JVM), os SGBD envolvidos no processo e as demais dependências que o projeto ou as tecnologias envolvidas possam ter. O presente trabalho propõe, também, um software web que vai abstrair toda complexidade de dependências que o processo de migração possa ter, deixando a cargo do próprio software;
- Com o software web, será possível processar uma migração independente de máquinas ou do hardware, bastando ter acesso à internet;
- Também, propõe-se a criação de *templates* (configurações) de migração, que podem ser reaproveitados quando tiver um processo de migração com esquemas previamente migrados, o que acarreta em velocidade no processo.

3.3 Migração entre sistemas gerenciadores de banco de dados

Mello (2009) apresenta duas etapas em sua parte prática, sendo a primeira a definição de um modelo de apoio para ser usado por softwares de migração e a segunda, uma ferramenta que utiliza esse modelo apresentado pelo autor.

O modelo consiste basicamente em um levantamento da estrutura das tabelas, os tipos de atributos, e as restrições do banco de origem. Após o levantamento dessas

informações é gerado um arquivo com instruções SQL contendo a estrutura previamente analisada para que se possam rodar essas instruções no banco de dados de origem.

Em seguida, há a geração de um novo arquivo de texto, também com instruções SQL, porém dessa vez direcionadas à inserção de dados, de modo que, tendo esse arquivo pronto, roda-se ele na base de destino para ter acesso aos dados. Na segunda etapa, o autor desenvolve uma aplicação na linguagem de programação Java para utilizar o modelo antes apresentado. Assim, visualizando o modelo apresentado pelo autor, e fazendo algumas modificações, foi elaborada uma proposta de *Minimum Viable Product* (MVP) do presente trabalho.

3.4 Requisitos para ferramentas de migração de dados

Santos Neto, Rodrigues Neto, Ribeiro Junior e Oliveira (2013) evidenciam alguns requisitos que devem ser atendidos por ferramentas de migração de dados, buscando trazer a satisfação de seus usuários, além de um processo de migração de dados realizado com sucesso. Os autores apresentam, nessa perspectiva, os seguintes requisitos:

- Reengenharia de dados;
- Migração de dados de diferentes fontes (CSV, banco de dados etc.);
- Processamento paralelo e distribuído;
- Migração incremental;
- Migração envolvendo paradigmas diferentes (Relacional, NoSQL);
- Migração de dados entre sistemas com modelagens diferentes (por exemplo, partir uma tabela em duas);
- Testabilidade dos dados migrados;
- Boa usabilidade.

Além de elencar esses requisitos, o artigo aplica eles a seis ferramentas de migração existentes no mercado, sendo elas:

- MySQL Migration Toolkit;
- JExodus;
- SwisSQL;
- DBConvert;
- DTM Migration Tool;

- Data Loader.

Conforme os testes práticos apresentados no artigo, algumas ferramentas são melhores em alguns aspectos e outras em outros, ficando a cargo do usuário mapear e escolher a ferramenta que melhor se adapte à sua necessidade. Nesse viés, os autores concluem o artigo comentando que há aspectos não atendidos por nenhuma ferramenta, além de outros a serem melhorados, abrindo espaço para a existência de novas ferramentas que cumpram com os requisitos apresentados.

O trabalho de Santos Neto, Rodrigues Neto, Ribeiro Junior e Oliveira (2013) auxiliou no entendimento dos requisitos que devem compor um sistema de migração de dados e no estabelecimento de um comparativo entre ferramentas que já estão no mercado. Isso auxiliou na construção de conhecimento sobre ferramentas que fazem migração de dados e sobre quais os pontos a serem melhorados.

3.5 Data Migration from Legacy Systems to Modern Database

Velimeneti (2016) apresenta um *case* prático do sistema de migração de dados de uma operadora da área da saúde, que utiliza um sistema de *mainframe* para armazenar seus dados, porém, com o passar do tempo esse sistema (por ser uma tecnologia antiga e não mais largamente utilizada) torna-se caro e apresenta problemas de inconsistência e de imprecisão. O autor, então, menciona funções, técnicas e metodologias para um processo de migração de um sistema antigo para novas tecnologias, por exemplo, usando o banco de dados SQL Server.

O presente estudo aproveitou-se das questões levantadas pelo estudo de caso de Velimeneti (2016), que teve como resultado algumas perguntas, por exemplo: qual a necessidade de as organizações migrarem seus dados? Quais os benefícios de um processo de migração para os donos de negócios? Qual o percentual de melhoria de desempenho do sistema após a migração? Conforme resultados positivos apresentados pelo autor em seu trabalho a esses questionamentos, evidencia-se a melhoria que um processo de migração pode conferir a uma organização.

4 METODOLOGIA

Este estudo apresenta como metodologia principal a pesquisa experimental, que se utiliza de procedimentos experimentais do processo de migração de dados. Este é o método mais prestigiado no meio científico e consiste, essencialmente, em definir um objeto de estudo, selecionar as variáveis que são capazes de modificá-lo e, então, escolher uma forma de controlar ou observar como essas variáveis produzem seus efeitos sobre o objeto (GIL, 2017).

Ainda conforme Gil (2017), o ator desse tipo de experimento passa de passivo para ativo, ou seja, deixa de ser um mero observador para agir e relatar os resultados do experimento, sendo que, para isso, o ator precisa modificar algum aspecto que julga ser responsável pela ocorrência de determinado fenômeno.

Tendo o tempo como variável a ser analisada, vai-se mensurar se a ferramenta de migração de dados proposta neste trabalho é capaz de reduzir o tempo necessário em um processo de migração de dados. Por fim, faz parte da pesquisa experimental deste estudo o levantamento dos requisitos necessários para elaboração da ferramenta de migração.

Ainda apresenta-se uma pesquisa bibliográfica, pois utiliza-se de materiais previamente publicados para fundamentar a teoria e os procedimentos utilizados para o desenvolvimento do trabalho. Nesse sentido, Gil (2017) diz que a pesquisa bibliográfica é elaborada com base em materiais já publicados, isto é, artigos, jornais, revistas, bem como materiais disponibilizados na internet. Esse tipo de pesquisa também tem a função de fornecer uma fundamentação teórica para o trabalho, além de apresentar qual o conhecimento adquirido para a elaboração do tema proposto.

Esta pesquisa também é qualitativa e, segundo Roesch (2013), apresenta-se com um viés que almeja medir a relação entre duas variáveis (causa-efeito), ou também mensurar o resultado de um projeto/sistema e, para isso, recomenda-se uma

boa estratégia de aferição, a fim de garantir uma boa interpretação dos resultados que se quer apresentar. São exemplos de estudos usando essa metodologia aqueles com viés experimental, comparando-se “um antes e um depois”, ou podendo utilizar séries temporais. Além disso, este trabalho apresenta também um caráter quantitativo, pois pretende-se mensurar o tempo que leva uma migração de dados, utilizando métodos convencionais (sem uso de ferramentas) e também uma ferramenta, já proposta neste estudo.

4.1 Ferramentas Utilizadas

O presente trabalho faz uso de dois SGBD: MySQL e PostgreSQL. A escolha do PostgreSQL se dá por sua alta popularidade, pois, como menciona Hong (2018, p. 8), ele é um dos SGBD de código aberto (disponível a qualquer pessoa) mais baixados e usados no mundo inteiro, sendo quase impossível estimar quantas pessoas o usam, uma vez que a maioria das *distros* (distribuições) Linux vem com esse SGBD instalado por padrão. O contador Linux estima que, atualmente, existem mais de 165.000 máquinas que rodam esse sistema operacional com mais de 600.000 usuários conforme última consulta em 14 de março de 2018.

Ainda conforme Hong (2018), o PostgreSQL é muito poderoso, tendo mais de 15 anos de desenvolvimento ativo com uma estrutura forte que lhe rendeu uma boa reputação de confiabilidade, integridade e proteção de dados. Além disso, ele também é compatível com os maiores sistemas operacionais da atualidade como Windows, Linux, Mac OS X e Unix.

Além disso, como ensina Milani (2006), o MySQL é um banco de dados completo e muito robusto, com um conjunto de características que permitem que ele seja usado no mercado. Nesse sentido, seus ambientes de utilização podem ser tanto acadêmicos como comerciais. Os testes e a ferramenta foram desenvolvidos em um notebook Acer com processador i7 de 8ª geração, com memória de 8GB e 128GB de SSD + 1 TB HDD.

As demais ferramentas para elaboração do sistema de migração de dados foram utilizadas porque atendem aos requisitos técnicos do projeto, além da familiaridade do autor com elas. Para elencá-las, apresenta-se o Quadro 3, contendo cada uma das ferramentas.

Quadro 3 - Ferramentas utilizadas no processo de desenvolvimento

Nome	**Versão**	**Descrição**
VS Code	1.44	IDE de desenvolvimento
PHP	7.4	Linguagem de programação
Laravel	9	Framework *back-end*
Angular	10	Framework *front-end*
PostgreSQL	12	Banco de dados
MySQL	5.7	Banco de dados
Docker	19.03	Plataforma de contêineres.

Fonte: Elaborado pelo autor (2020).

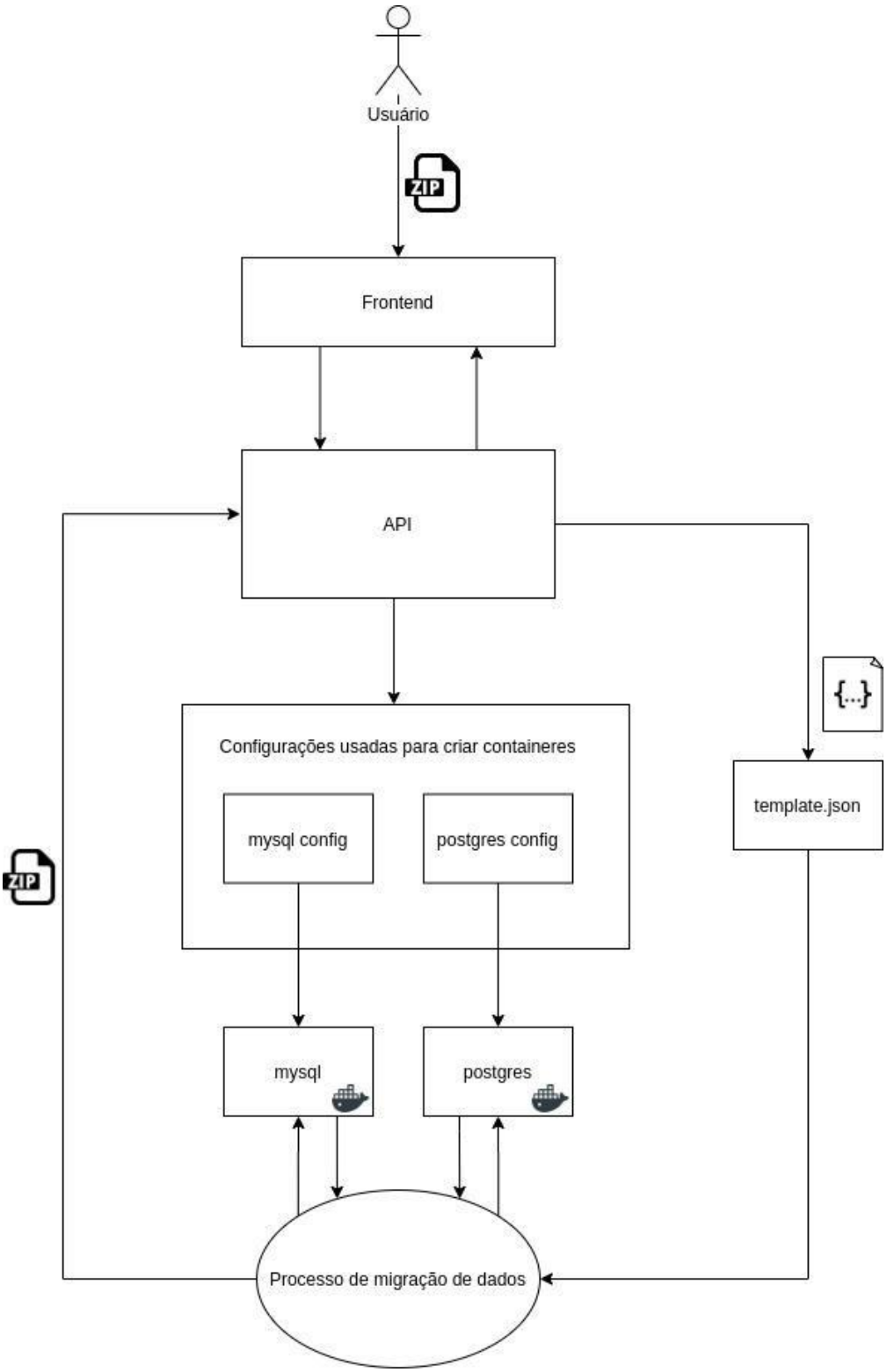
5 ESPECIFICAÇÃO DO PROJETO

Este capítulo discorre sobre como a ferramenta opera, além de apresentar os requisitos necessários para a elaboração da ferramenta e da estrutura tecnológica envolvida.

5.1 Visão geral

Para entender o funcionamento do software em seu processo completo de migração, é apresentado um descritivo da estrutura proposta para a solução de migração de dados na Figura 4, a seguir. A imagem serve como ilustração do processo e será explicada nos parágrafos que a sucedem.

Figura 4 - Estrutura do software de migração



Fonte: Elaborado pelo autor (2020).

Na Figura 4, o usuário representa o operador do sistema, que envia ao *front-end* da aplicação os dois arquivos zip necessários (com os *dumps* dos BD), o banco de origem e o de destino, além de cumprir com uma série de passos que será responsável por gerar o arquivo JavaScript Object Notation (*template* em formato JSON), que será usado pela API no processo de migração para saber como lidar com os dados enviados pelo usuário.

Além de interagir com o usuário, a API será responsável por gerir os containers usados no processo de migração, gerando-os conforme a necessidade para aquele processo e criando os bancos de dados de acordo com os arquivos zip enviados. Em seguida, a API usará os bancos de dados criados nos contêineres e o *template* criado a partir das informações fornecidas pelo usuário para efetuar a migração e, ao final, destruir os containers do processo, disponibilizando um arquivo .tar com a base migrada para o usuário.

5.2 Requisitos

Segundo Vazquez e Simões (2016), requisito pode ser definido como uma condição ou necessidade de um usuário que serve para resolver um problema ou alcançar um determinado objetivo, ou uma condição que deve ser atendida por um componente de software ou de hardware para cumprir com o disposto em um documento formalmente definido. Já para Kerr (2015) requisitos de um sistema são as especificações daquilo que o sistema deve fazer, os serviços que vai oferecer e as restrições do seu funcionamento.

Os requisitos do projeto foram propostos pelo autor do trabalho e baseados em demandas de uma empresa em que ele atuou e que continha um ERP no ramo de seguros, cujos casos de migração de banco de dados eram recorrentes e sofriam sempre um processo árduo e lento por não conter uma ferramenta especializada e específica para migração de dados.

5.3 Requisitos funcionais

Os requisitos funcionais têm a função de descrever o comportamento do software na visão do usuário em termos de tarefas e serviços, ao contrário de especificações como linguagens de programação, plataformas e bibliotecas

(VAZQUEZ; SIMÕES, 2016). O Quadro 4 apresenta os requisitos funcionais do projeto, contendo um atributo-código identificador único do requisito, sua descrição e a prioridade que pode obter três valores Requisito Funcional Obrigatório (RF-O, que deverá ser obrigatoriamente implementado no projeto), Requisito Funcional Importante (RF-I, que é um requisito importante de ter no *software* ou em versões futuras, e o Requisito Funcional Desejável (RF-D, que foi elaborado verificando funcionalidades em outras ferramentas do mesmo ramo de atuação ou em bibliografia).

Quadro 4 - Especificação dos requisitos funcionais

Código	Descrição	Prioridade
RF-00001	Possibilita ver o *schema* dos banco de dados, tanto de origem quanto de destino.	RF-O
RF-00002	Permite correlacionar os campos da tabela de origem com os da tabela de destino para ser lido pela ferramenta de migração	RF-O
RF-00003	Salvar um *template* com as tabelas correlacionadas pelo operador no formato JSON para ser utilizado pela ferramenta	RF-O
RF-00004	Os tipos de dados detectados deverão ser tratados pela ferramenta	RF-O
RF-00005	Ao final do processo de migração deverá ser disponibilizado um *dump* com os dados e estrutura do banco migrado	RF-O
RF-00006	Migrar dados de diferentes fontes (CSV, Excel, arquivos em texto etc.)	RF-D
RF-00007	Possibilita a seleção de um *template* de migração para caso seja uma migração entre bancos já migrados alguma vez	RF-O
RF-00008	Migração envolvendo paradigmas diferentes (banco de dados relacional para bancos de dados NoSQL)	RF-D
RF-00009	Tem um processamento paralelo e distribuído	RF-I
RF-00010	Utiliza o padrão SQL ANSI para manter uma compatibilidade entre comandos SQL em diferentes SGBDs.	RF-O

Fonte: Elaborado pelo autor (2020).

5.4 Requisitos não funcionais

Os requisitos não funcionais complementam a especificação da ferramenta a ser desenvolvida, pois mais que funcionar, eles devem funcionar bem. Enquanto os

requisitos funcionais referem-se a tarefas de usuários, os não funcionais abordam restrições gerais ao software como a qualidade, a confiabilidade, a facilidade de uso e a manutenção facilitada (VAZQUEZ; SIMÕES, 2016).

Dito isso, definem-se os requisitos não funcionais no Quadro 5, elaborado da seguinte maneira: com um atributo-código identificador único do requisito, descrição desse requisito e a prioridade, que pode obter três valores, tais como Requisito Não Funcional Obrigatório (RNF-O, que deverá ser obrigatoriamente implementado no projeto); Requisito Não Funcional Importante (RNF-I, que é um requisito importante para se ter no software ou em versões futuras, e o Requisito Não Funcional Desejável (RNF-D, que foi elaborado verificando-se funcionalidades em outras ferramentas do mesmo ramo de atuação ou em bibliografia).

Quadro 5 - Especificação dos requisitos não funcionais

Código	Descrição	Prioridade
RNF-00001	Utiliza a linguagem de programação PHP no *back-end*	RNF-O
RNF-00002	Utiliza a linguagem de programação Javascript no *front-end*	RNF-O
RNF-00003	Utiliza os *frameworks* Laravel e Angular para *back-end* e *front-end*, respectivamente	RNF-O
RNF-00004	Utiliza Docker como plataforma para virtualizar os contêineres	RNF-O
RNF-00005	Utiliza os bancos de dados PostgreSQL e MySQL	RNF-O
RNF-00006	Tem uma API para gerenciar os contêineres e outra para o processo de migração de dados	RNF-D

Fonte: Elaborado pelo autor (2020).

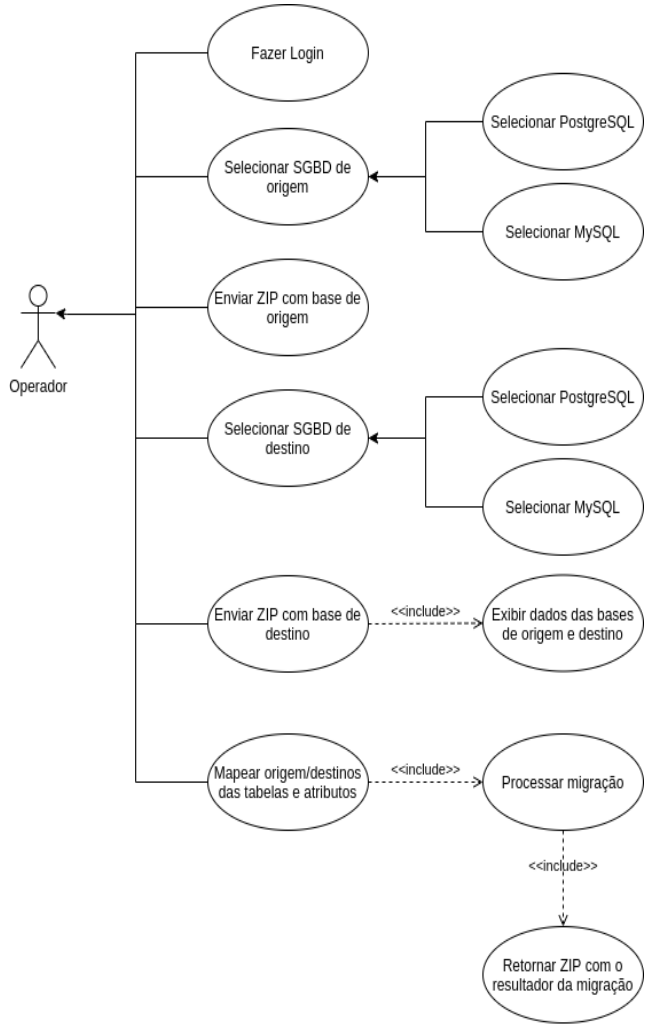
5.5 Caso de uso

Para Medeiros (2004), um caso de uso pode ser definido como uma macroatividade que contém, em seu interior, tarefas e atividades menores que, ao serem executadas, formam a macroatividade, isto é, esse tipo de atividade compreende a realização de várias ações que resultam no todo.

Para melhor exemplificar o processo de migração proposto neste trabalho, foi elaborado um diagrama de caso de uso, conforme apresentado na Figura 5. Nela, o operador é o usuário principal do software e, após a ação de “fazer login”, ele

necessita informar ao sistema qual o SGBD de origem, além de enviar uma cópia do banco de dados de origem (no formato .zip), repetindo o mesmo procedimento para o banco de dados de destino, a fim de que o sistema possa carregar em tela as tabelas e atributos a serem relacionados pelo operador (gerando o *template* de migração para esse processo). Posteriormente, o sistema analisará o *template* configurado pelo operador para processar a migração e retornará o resultado da migração em uma cópia (em formato .zip) ao operador.

Figura 5 - Caso de uso do software



Fonte: Elaborado pelo autor (2020).

5.6 Modelo entidade-relacionamento

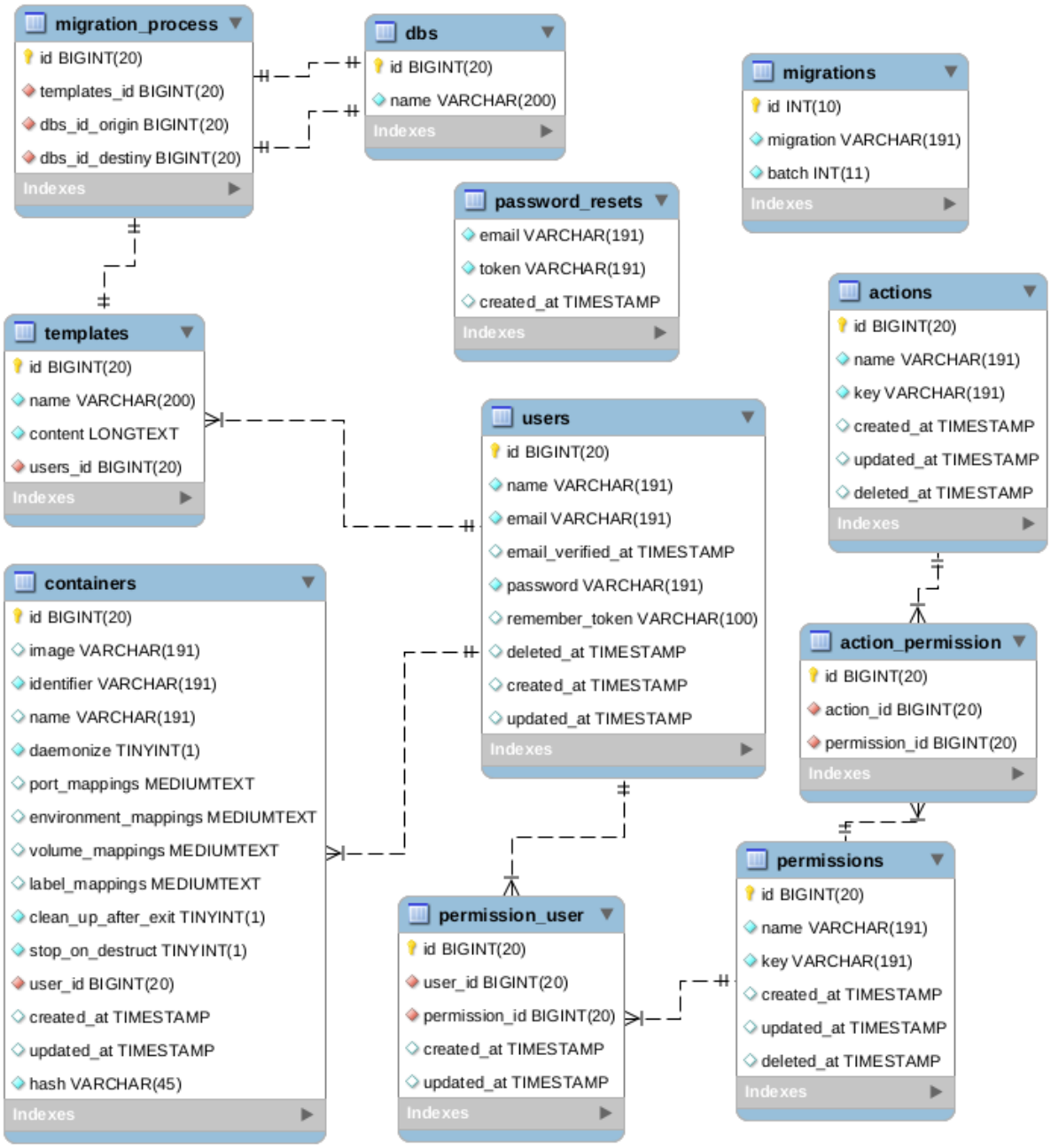
Para o presente trabalho foi elaborado um banco de dados para armazenamento dos processos de migração, controle de acessos do usuário, armazenamento de *templates* criados pelo usuário, iniciação e armazenamento dos

contêineres que estão sendo usados no processo de migração. Além disso, para demonstrar a estrutura do banco de dados, foi proposto um modelo entidade-relacionamento, cuja abordagem é proposta com base no modelo ER elaborado por Peter Pin-Shan Chen, no ano de 1976, sendo esse um aprimoramento e uma das técnicas de modelagem semânticas mais conhecidas (DATE, 2004).

De acordo com Elmasri e Navathe (2018), o modelo entidade-relacionamento pode ser definido como um modelo de dados conceitual de alto nível e suas variações são empregadas para a consolidação de um projeto de aplicação de banco de dados. Justificando a importância do modelo entidade-relacionamento, Elmasri e Navathe (2018) complementam, dizendo que as modelagens de objetos estão se tornando populares e especificando detalhes dos módulos de softwares com suas interações utilizando diagramas.

No modelo apresentado na Figura 6 temos as estruturas projetadas do banco de dados nos quadros *actions, actions_permissions, permissions, permission_user*, que são responsáveis por armazenar o esquema de permissões de acessos ao sistema. Já no quadro *users*, temos o armazenamento dos usuários, enquanto a tabela *migration_process* é responsável por armazenar um processo de migração inicializado pelo usuário e o quadro *templates* armazena os *templates* configurados pelo usuário, que serão utilizados em um processo de migração. Além disso, o quadro de *containers* armazena os containers envolvidos em um processo de migração, enquanto que o *dbs* contém os SGBD suportados pela plataforma. Ainda, os quadros *password_resets* e *migrations* são usadas pelo *framework* Laravel.

Figura 6 - Modelo entidade-relacionamento do software proposto



Fonte: Elaborado pelo autor (2020).

6 IMPLEMENTAÇÃO DA FERRAMENTA

Nesta seção, será apresentado o desenvolvimento da ferramenta, demonstrando as características do projeto, as telas da ferramenta, suas funcionalidades e partes do código. No presente estudo, como dito anteriormente, propõe-se o desenvolvimento de uma ferramenta de migração de dados para auxiliar pequenas e médias empresas que desejam migrar seus bancos de dados de um SGBD para outro.

Para alcançar os objetivos propostos neste estudo, foi utilizada a estratégia de migração *Big Bang*, em que é necessário parar o uso do banco de origem e destino no momento da migração e a metodologia de migração ETL (*Extract, Transform and Load*) que permite a extração dos dados, algumas transformações e a carga dos dados no banco de dados destino. Além disso, para avaliar a ferramenta foi proposto um cenário de migração de uma base de dados *Employees Database* disponibilizada no site da MySQL, na qual foi mensurado o tempo de migração utilizando a ferramenta desenvolvida e por meio de SQL sem auxílio de ferramentas de migração.

6.1 Estrutura e conexões

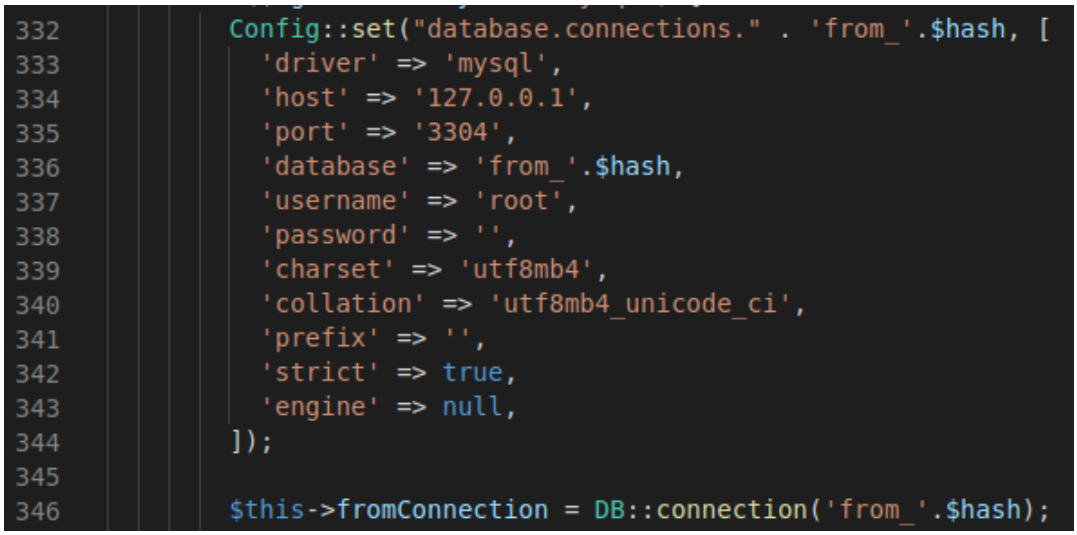
A ferramenta foi pensada para, em projetos futuros, adicionar novos SGBD ao processo de maneira facilitada. Para isso, as conexões com os bancos de dados são dinâmicas, assim como a disponibilização do serviço SGBD ser de fácil implementação em função do uso do Docker.

Na Figura 7 é possível ver como são elaboradas as conexões de banco de dados no processo de migração, sendo que essas configurações são utilizadas para conectar-se ao banco de dados de origem e obter as informações a serem migradas. Essas configurações são feitas durante a migração e, ao final do processo, a

ferramenta não mantém essas conexões (pois os containers que contém os SGBD são finalizados).

Também é possível verificar, na linha 336, que o nome do banco é dinâmico, sendo uma concatenação entre a palavra *from_* e uma *hash*, gerada dinamicamente no início do processo de migração e usada para identificar o processo de migração, por ser dinâmica e única em toda tabela.

Figura 7 - Configurando conexão com o banco



Fonte: Elaborado pelo autor (2020).

Na Figura 8, da linha 451 à linha 457, é criada uma instância nova do Docker, utilizando a imagem mysql:5.7, ao passo que nas linhas 452 e 453 são configuradas algumas variáveis utilizadas pelo container, sendo que a primeira permite que o SGBD aceite senhas em branco e a segunda é criada um banco de dados padrão. Na linha 454, há uma importante configuração de mapeamento de volume, o que permite haver uma pasta compartilhada entre o *host* (onde encontra-se a API) e o container, fazendo com que a API acesse o resultado da migração (*dump* SQL). Já na linha 456 é definido o nome do container.

Na linha 459, por sua vez, são obtidas as configurações da instância *docker* recém-criada, para que da linha 460 até a 473 sejam inseridas essas informações no banco de dados.

Figura 8 - Instância docker e configuração do container

```
451     $this->mysqlInstance = DockerContainer::create('mysql:5.7')
452       ->setEnvironmentVariable('MYSQL_ALLOW_EMPTY_PASSWORD', true)
453       ->setEnvironmentVariable('MYSQL_DATABASE', 'teste')
454       ->setVolume($hostPath, '/dump')
455       ->mapPort('3304', '3306')
456       ->name('imigration_mysql_'.$hash)
457       ->start();
458
459     $mysqlConfigs = (array) $this->mysqlInstance->getConfig();
460     Container::create([
461       'image' => 'mysql:5.7',
462       'identifier' => $this->mysqlInstance->getShortDockerIdentifier(),
463       'name' => $this->mysqlInstance->getName(),
464       'daemonize' => ! ! $mysqlConfigs['daemonize'],
465       'port_mappings' => json_encode($mysqlConfigs['portMappings']),
466       'environment_mappings' => json_encode($mysqlConfigs['environmentMappings']),
467       'volume_mappings' => json_encode($mysqlConfigs['volumeMappings']),
468       'label_mappings' => json_encode($mysqlConfigs['labelMappings']),
469       'clean_up_after_exit' => ! ! $mysqlConfigs['cleanUpAfterExit'],
470       'stop_on_destruct' => ! ! $mysqlConfigs['stopOnDestruct'],
471       'user_id' => Auth::id(),
472       'hash' => $hash
473     ]);
```

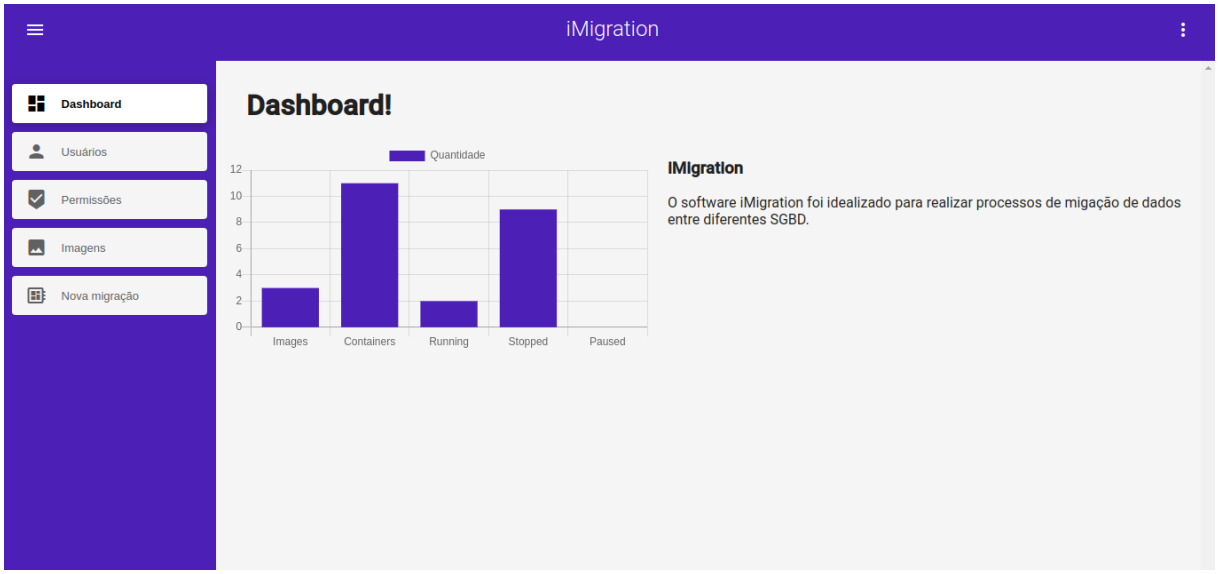
Fonte: Elaborado pelo autor (2020).

6.2 Telas da ferramenta

Este capítulo apresenta uma visualização geral da ferramenta para ambientar o leitor à interface que foi implementada. A esse respeito, a Figura 9 apresenta uma visualização geral da ferramenta, em que, logo após entrar no sistema, é acessada a tela de *dashboard*, que apresenta um gráfico com informações sobre os containers e seus estados: rodando, parado ou pausado, o que pode ser usado como consulta pelo usuário.

Ao lado esquerdo, temos o menu com as telas de usuário, usadas para inserir, editar e/ou deletar usuários; a tela de permissões, que contém as permissões dos usuários; uma tela de imagens, que contém a exibição de imagens *dockers* disponíveis na API e a tela de nova migração.

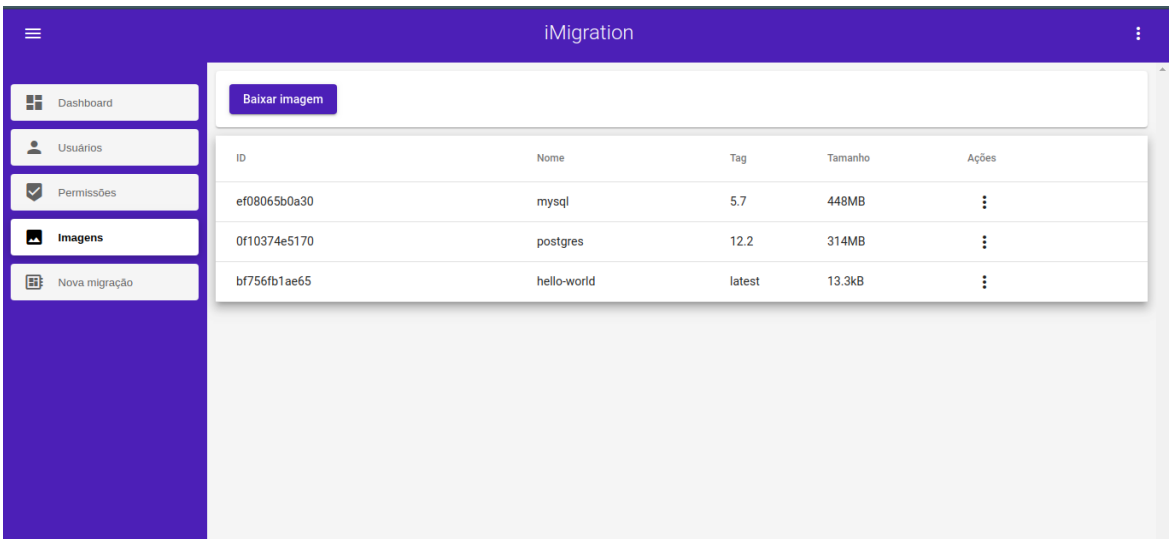
Figura 9 - Tela inicial da ferramenta



Fonte: Elaborado pelo autor (2020).

Foi elaborada também uma tela de listagem de imagens *dockers* disponíveis no servidor. Além disso, há a possibilidade de baixar as imagens direto do *docker hub* visando a uma futura evolução da ferramenta.

Figura 10 - Tela inicial de listagem das imagens

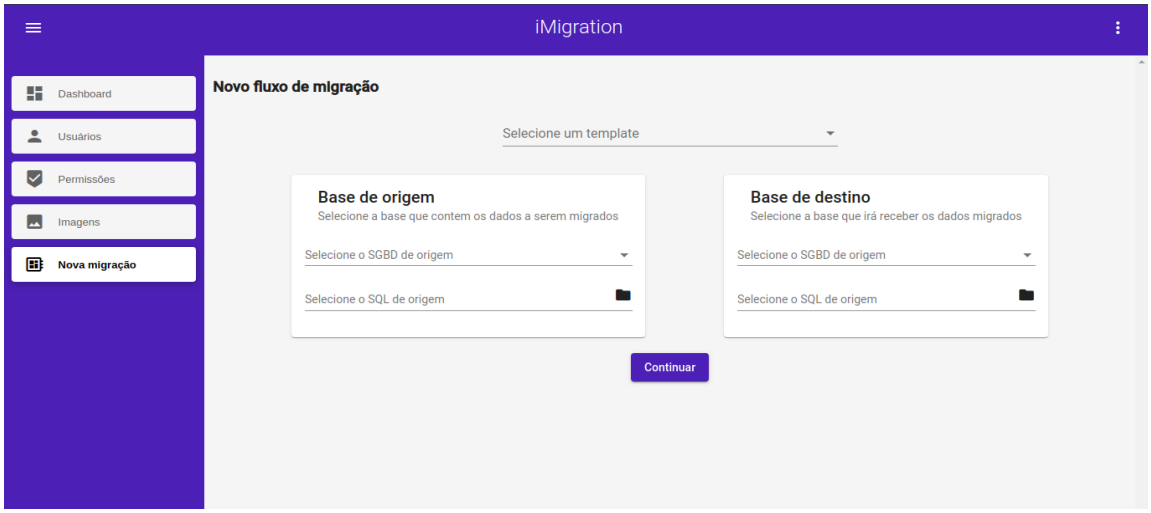


Fonte: Elaborado pelo autor (2020).

O processo de migração está disponível na tela de “Nova migração”, na Figura 11, que mostra a primeira tela de configuração do fluxo. O primeiro campo da tela

(Selecione um template), permite que o operador escolha um *template* criado previamente em outras migrações ou selecione a opção “Criar novo template” para criar um novo *template* para a migração atual.

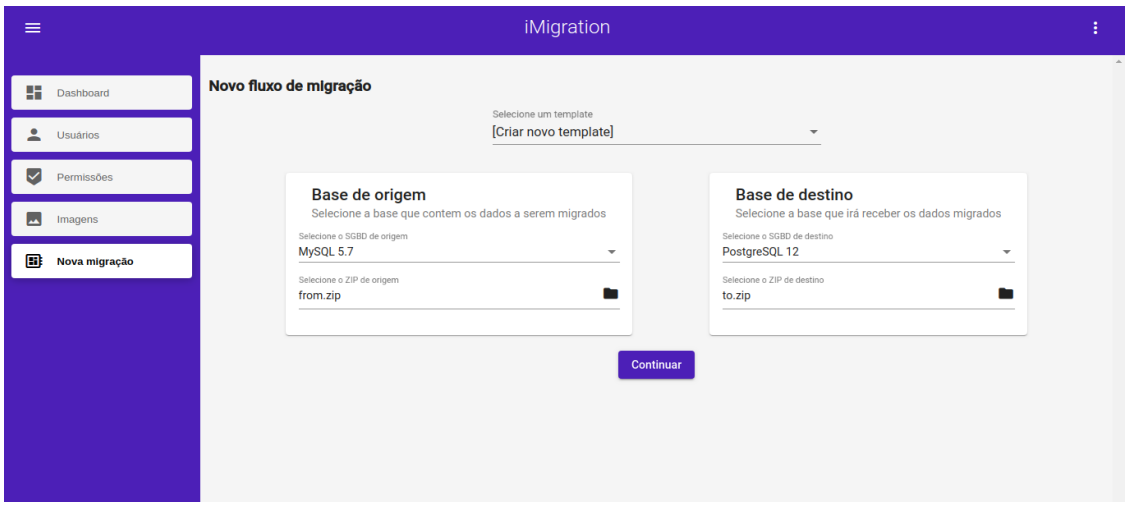
Figura 11 - Tela de novo fluxo de migração



Fonte: Elaborado pelo autor (2020).

No cartão de origem, o operador deve selecionar o SGBD de origem, que contém as opções MySQL 5.7 e PostgreSQL 12 e, após, enviar um .zip contendo o *dump* do banco de dados. Já no cartão de destino, deve-se fazer o mesmo processo, sendo que o .zip deve conter um *dump* da base de destino (que receberá os dados). A Figura 12 exemplifica como fica a tela com os campos preenchidos.

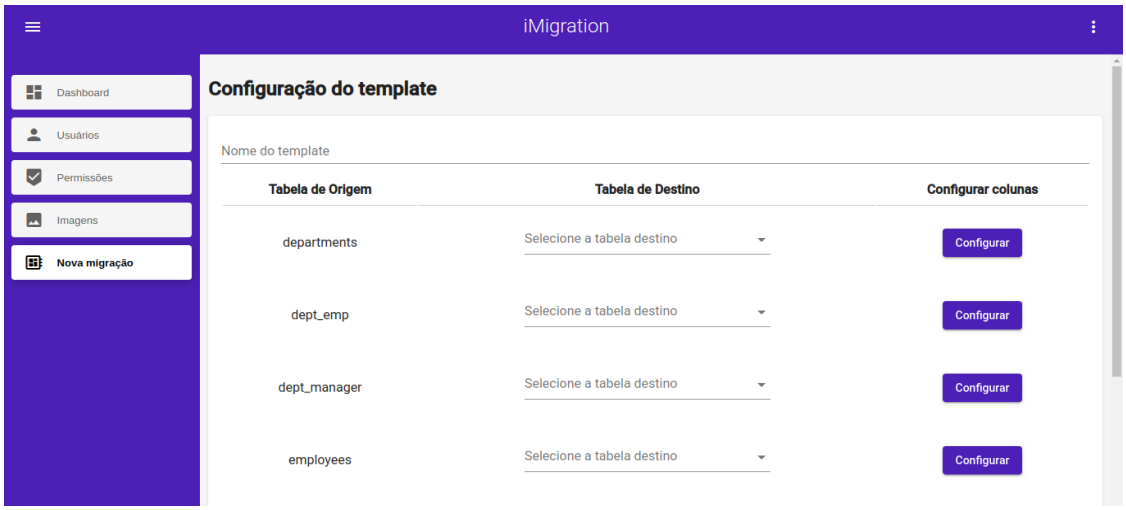
Figura 12 - Exemplo da tela de novo fluxo de migração



Fonte: Elaborado pelo autor (2020).

A API recebe os *dumps* e as informações gerando os containers para migrar esses dados e, no exemplo supracitado, como foi selecionada a opção de criar novo *template*, a API consulta os metadados dos dois bancos de dados e retorna as tabelas e colunas de ambos, para que o operador possa correlacioná-los. A Figura 13, por sua vez, apresenta essa relação entre tabelas e colunas, o que irá gerar o *template* de migração para esse processo.

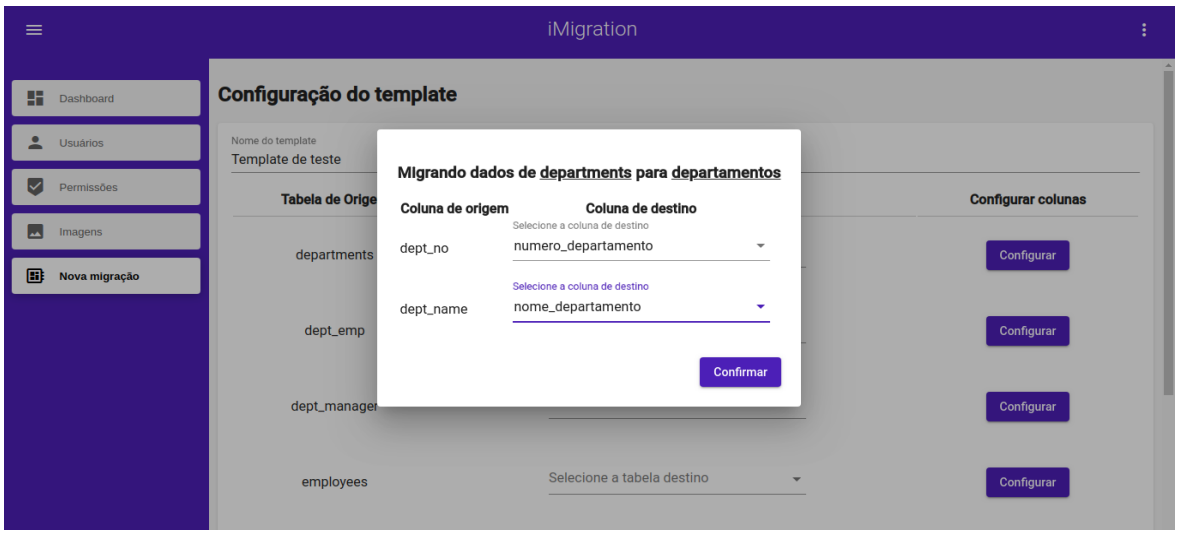
Figura 13 - Tela de configuração de template



Fonte: Elaborado pelo autor (2020).

Nesta tela, o primeiro campo é o nome do *template*. Em seguida, é montada uma listagem em que a “Tabela de Origem” indica as tabelas lidas do SGBD de origem e as “Tabela de Destino” contêm um campo de seleção em que o usuário seleciona a tabela de destino. Além disso temos uma opção de “Configurar colunas” que é usada para correlacionar as colunas da tabela. A esse respeito, a Figura 14 demonstra como são configuradas as colunas para esse processo.

Figura 14 - Configuração das colunas

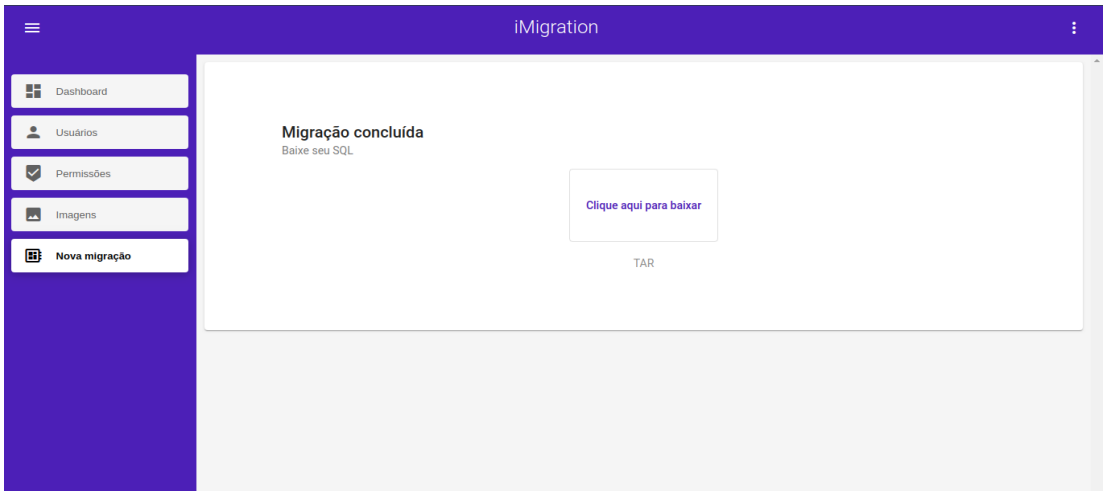


Fonte: Elaborado pelo autor (2020).

Para geração do template, serão consideradas apenas as tabelas que foram selecionadas, ou seja, se o operador não selecionar uma tabela de destino para uma determinada tabela de origem, ela não será migrada. Além disso, a configuração desse *template* permite que, na próxima migração desse mesmo banco de dados, o operador apenas selecione o template na tela inicial, conforme apresentado na Figura 11, podendo repetir essa migração quando for necessário.

Quando a migração for concluída, será disponibilizado ao usuário um arquivo em formato .tar com o resultado da migração, que pode ser acessado clicando-se no botão “Clique aqui para baixar”, como apresentado na Figura 15.

Figura 15 - Disponibilizando resultado da migração para o usuário



Fonte: Elaborado pelo autor (2020).

Ainda, na Figura 15, o operador já tem acesso ao resultado final da migração, podendo restaurar o *dump* eu seu SGBD para ter acesso aos dados migrados.

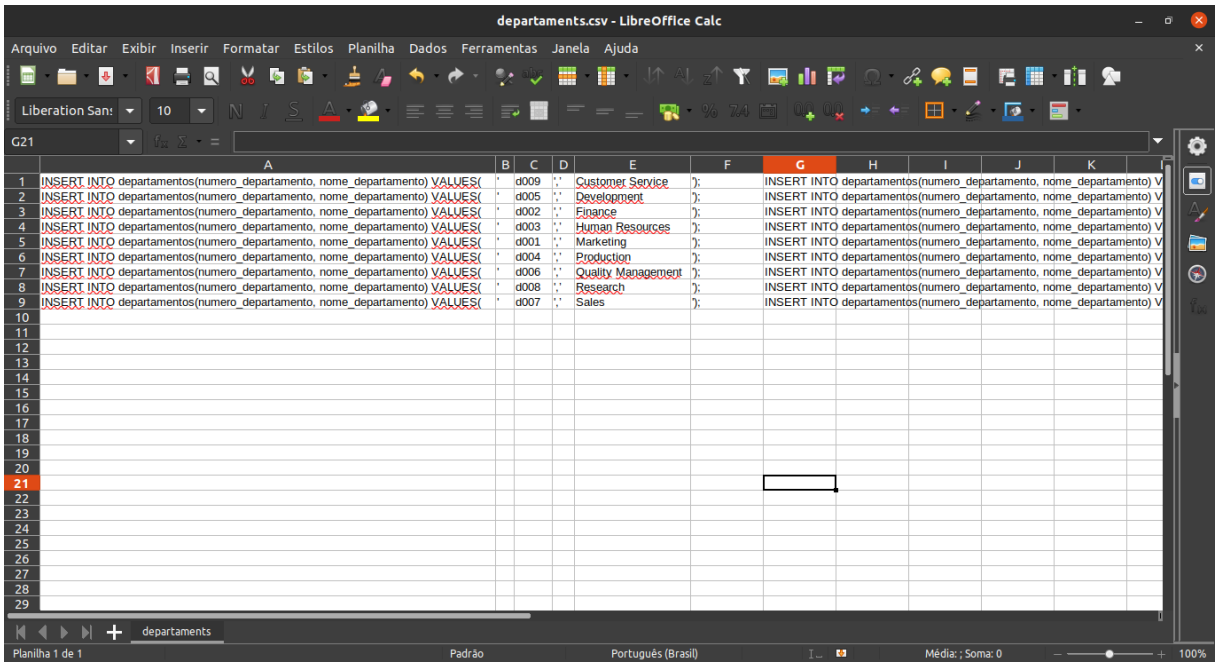
7 RESULTADOS

Como objetivo principal, a ferramenta reduz a complexidade do processo de migração e diminui o tempo necessário para uma migração segura e eficaz, uma vez que processos de migração não automatizados tendem a ter diversos problemas, como dados inseridos incorretamente, migrações longas e cansativas, além de muitas vezes o processo de migração ser repetitivo, pois ocorre do mesmo software de origem para o mesmo software de destino (caso muito comum em empresas que desenvolvem softwares).

Para minimizar essa migração repetitiva de uma mesma origem para um mesmo destino, a ferramenta proposta neste trabalho apresenta um conceito de *templates* configurados apenas na primeira migração e que depois podem ser reutilizados (caso seja de uma mesma origem e destino), o que confere agilidade ao processo de migração.

A migração manual, utilizando SQL, levou um total de três horas e quarenta e cinco minutos com um processo muito mais trabalhoso e detalhista do que o processo feito pela ferramenta. A primeira etapa da migração via SQL consistiu em exportar os dados da tabela para um arquivo CSV e, na sequência, esse arquivo foi aberto em um software de manipulação de dados tabulados, de modo que, através dele, foram montados os comandos SQL para serem importados na base de destino. A Figura 16 demonstra a montagem dos comandos SQL.

Figura 16 - Montagem dos comandos SQL



Fonte: Elaborado pelo autor (2020).

É possível verificar que, na coluna G, os comandos estão na linguagem SQL. Esses comandos foram copiados para um arquivo e salvos no formato SQL, para que depois possam ser importados na base de destino.

Figura 17 - Arquivo no formato SQL

```
1 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10001','1986-06-26','9999-01-01');
2 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10002','1996-08-03','9999-01-01');
3 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10003','1995-12-03','9999-01-01');
4 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10004','1986-12-01','9999-01-01');
5 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10005','1989-09-12','9999-01-01');
6 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10006','1990-08-05','9999-01-01');
7 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10007','1989-02-10','9999-01-01');
8 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10008','1998-03-11','2000-07-31');
9 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10009','1985-02-18','9999-01-01');
10 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10010','2000-06-26','9999-01-01');
11 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10011','1990-01-22','1996-11-09');
12 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10012','1992-12-18','9999-01-01');
13 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10013','1985-10-20','9999-01-01');
14 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10014','1993-12-29','9999-01-01');
15 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10015','1992-09-19','1993-08-22');
16 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10016','1998-02-11','9999-01-01');
17 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10017','1993-08-03','9999-01-01');
18 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10018','1992-07-29','9999-01-01');
19 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10019','1999-04-30','9999-01-01');
20 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10020','1997-12-30','9999-01-01');
21 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10021','1988-02-10','2002-07-15');
22 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10022','1999-09-03','9999-01-01');
23 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10023','1999-09-27','9999-01-01');
24 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10024','1998-06-14','9999-01-01');
25 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10025','1987-08-17','1997-10-15');
26 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10026','1995-03-20','9999-01-01');
27 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10027','1995-04-02','9999-01-01');
28 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10028','1991-10-22','1998-04-06');
29 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10029','1999-07-08','9999-01-01');
30 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10030','1994-02-17','9999-01-01');
31 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10031','1991-09-01','9999-01-01');
32 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10032','1990-06-20','9999-01-01');
33 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10033','1987-03-18','1993-03-24');
34 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10034','1995-04-12','1999-10-31');
35 INSERT INTO departamento_empregado_ultima_data(numero_empregado, de_data, ate_data) VALUES('10035','1988-09-05','9999-01-01');
```

Fonte: Elaborado pelo autor (2020).

Para tabelas com muitos dados, como era o caso da tabela “*salaries*”, que continha aproximadamente dois milhões de registros, foi necessária a quebra em

vários arquivos SQL contento, no máximo, quatrocentos mil registros, pois era o limite que o Libre Office conseguia abrir em quantidade de linhas.

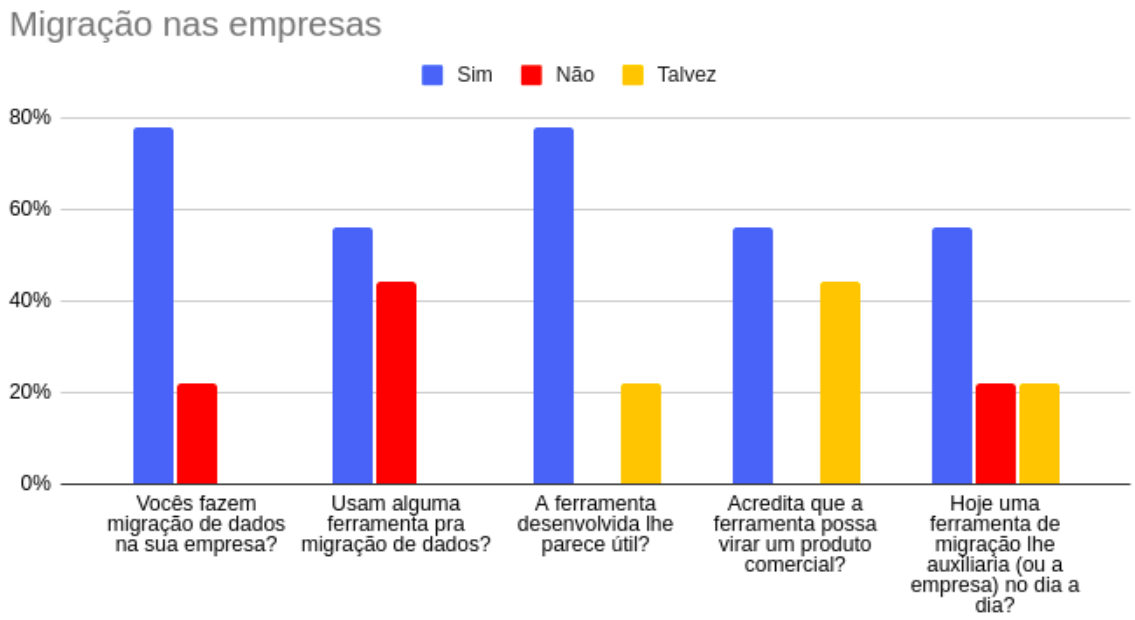
Para validação do tempo de migração utilizando a ferramenta proposta neste trabalho, foram escolhidos dois testes utilizando a base de dados *Employees sample database.* Primeiramente, foi feita uma migração contendo uma leitura (*buffer*) de 1.000 registros por vez da base de origem e outra com 50.000 registros por vez. Esses registros foram usados para compor um vetor de dados, visto que essas quantias precisam ser definidas para que o software não copie todos os dados do banco direto para a memória, ocasionando um estouro de pilha.

A migração utilizando 1.000 registros levou 01h48min (uma hora e quarenta e oito minutos) enquanto que a migração utilizando 50.000 registros levou 02h11min (duas horas e onze minutos). Ao final do processo, foi validada a quantidade de linhas da base de origem e de destino, como forma de garantir que todos os registros foram migrados.

Como forma de testar a usabilidade da ferramenta, o presente trabalho propôs um breve questionário, que foi aplicado a nove desenvolvedores, aos quais vamos nos referir como participante 1, participante 2, participante 3 e assim sucessivamente. O questionário teve por objetivo obter a opinião dos desenvolvedores acerca da ferramenta desenvolvida e contém oito perguntas objetivas com as opções “Sim”, “Não” e “Talvez”. Ao final do questionário, há um campo em que os participantes podem deixar algum comentário sobre a ferramenta.

Dos participantes, cinco trabalham em empresas na cidade de Lajeado, três em empresas residentes em Porto Alegre e um em uma empresa de Teutônia. Uma empresa é de grande/médio porte, três delas são de médio porte e cinco de pequeno porte. A Figura 18, a seguir, apresenta os resultados em percentuais das respostas dos participantes sobre os questionamentos referentes à migração nas empresas em que eles trabalham.

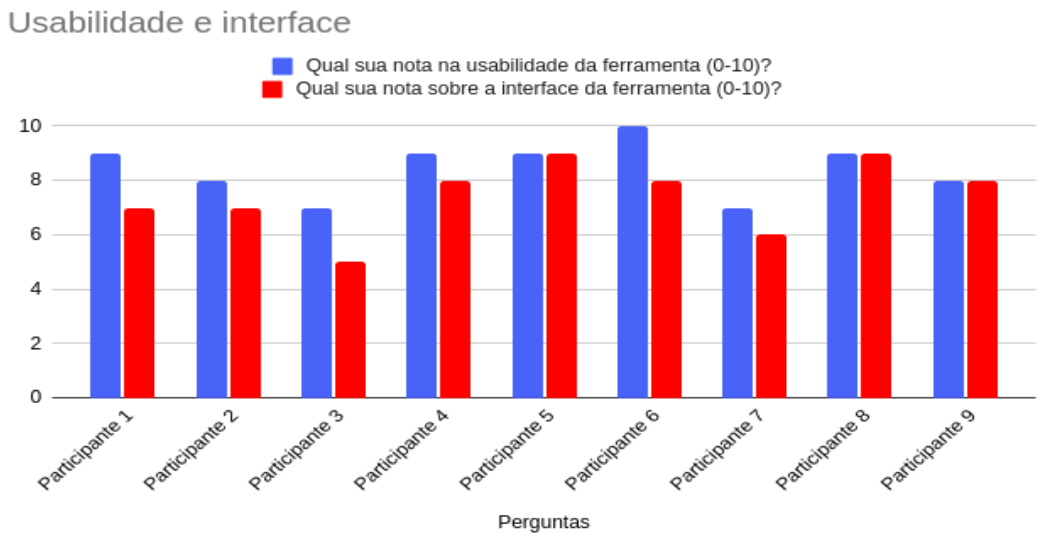
Figura 18 - Gráfico de migração nas empresas



Fonte: Elaborado pelo autor (2020).

Já na Figura 19 temos os resultados de usabilidade e de interface. Eles foram separados em um novo gráfico, pois os participantes deveriam dar uma nota de 0 a 10 sobre a usabilidade e a interface da ferramenta.

Figura 19 - Gráfico de usabilidade e interface



Fonte: Elaborado pelo autor (2020).

No final do questionário, os participantes podiam deixar uma opinião sobre a ferramenta, sendo que três deles deixaram comentários. O primeiro relatou que poderia haver mais opções de SGBD para ampliá-la, enquanto um outro relatou que, com o avanço da tecnologia e o uso de soluções de *cloud* (nuvem) e *serverless* (servidores como serviço), as plataformas têm desenvolvido soluções de migração próprias como a AWS (*Amazon Web Services*), que contém a ferramenta *AWS Database Migration Service*. Já o último comentário aconselha que a ferramenta tenha um diferencial das que já existem, para que se destaque e possa ganhar espaço no mercado.

8 CONSIDERAÇÕES FINAIS

Na construção da ferramenta, foram enfrentados alguns problemas em relação à execução de rotinas diretas no sistema operacional e de rotinas como extrair os *dumps* enviados pelo usuário, o que se tornou um problema, uma vez que na linguagem PHP não temos funções de *callbacks* (funções de execução, após determinado evento). Dessa forma, esses problemas tiveram que ser resolvidos com funções como a *sleep* (aguardar). Inicialmente, o usuário enviava os *dumps* dos bancos de dados em formato SQL, que logo foi descartado devido ao tamanho das bases de dados, o que acarretou em estouro do tempo de execução para envio de arquivos ao servidor e, portanto, foi migrado para o formato .zip (compactado).

O processo de migração mostrou-se mais rápido, ágil e seguro utilizando a ferramenta desenvolvida, sendo que no processo automatizado com o uso da ferramenta foram percebidas diversas vantagens, tais como:

- **Tempo:** o operador leva menos tempo migrando pela ferramenta do que fazendo a migração via SQL (manualmente);
- **Agilidade:** o processo automatizado permite que o operador faça outros trabalhos enquanto aguarda a migração ser completada, ao passo que no formato manual, o operador precisa estar dedicado somente a atividade;
- **Reaproveitamento de migrações:** caso haja outra migração de mesma origem e destino, o usuário pode utilizar o template criado na primeira migração;
- **Segurança:** uma vez que os dados estão sendo inseridos via um processo automatizado, a chance de erros diminui;
- **Simplicidade:** a migração pela ferramenta é mais simples de ser feita, pois exige menos processos.

Os resultados do questionário aplicado a profissionais que já trabalham na área reforçam a aplicabilidade de uma ferramenta dessa natureza, porém voltada a pequenas e médias empresas, pois médias/grandes e grandes empresas tendem a usar serviços das próprias plataformas em que hospedam seus códigos. Além disso, a ferramenta de migração é útil para empresas que contêm processo de migração manual, pois as notas para a ferramenta foram maiores na empresa a qual a migração era manual.

Conclui-se, então que a ferramenta obteve êxito ao cumprir o proposto neste trabalho, estando limitada ao escopo dele. Porém, como migração é um assunto recorrente e importante nas empresas, é possível tornar essa ferramenta comercial e, pensando nisso, os trabalhos futuros apresentam ajustes e melhorias que podem ser efetuados para tornar a ferramenta mais comercial.

8.1 Trabalhos futuros

A ferramenta desenvolvida não tem fins comerciais e foi projetada pensando nos problemas levantados por este estudo, porém ela foi pensada para ser adaptável e adicionar novos SGBD com facilidade ao processo. No decorrer do estudo deste trabalho, assim como na prática do seu desenvolvimento, foram elencados aprimoramentos que podem servir como forma de tornar o produto mais completo:

- **Migração utilizando *multithreads*:** um processo de migração de dados utilizar *multithreads*, que podem gerar um ganho significativo nos tempos de execução;
- **Utilizar *call-backs*:** a utilização de funções de *call-back* se mostra muito vantajosas em processos de chamadas de rotinas do sistema operacional;
- **Cliente para envio dos *dumps*:** ter um cliente (software) que envia os *dumps* pode ser vantajoso, uma vez que banco de dados muito grandes não poderão utilizar a ferramenta devido à limitação de tamanho de upload de arquivo;
- Migração de banco de dados não relacionais;
- **Criar containers com SGBD dinâmicos:** operador criar containers contendo o SGBD necessário para a migração de dados.

REFERÊNCIAS

ALVES, W. P. **Banco de dados**. São Paulo: Saraiva, 2014.

AMADEU, C. V. **Banco de dados**. São Paulo: Pearson, 2015. Disponível em: https://plataforma.bvirtual.com.br/Leitor/Publicacao/22152/pdf/0. Acesso em: 04 abr. 2020.

AUDY, J. L.; ANDRADE, N.; CIDRAL, A. G. K.. **Fundamentos de sistemas de informação**. Porto Alegre: Bookman, 2007. Disponível em: https://integrada.minhabiblioteca.com.br/#/books/9788577801305/cfi/2!/4/4@0.00:59.7. Acesso em: 29 mar. 2020.

BATISTA, E. O. **Sistemas de Informação:** o uso consciente da tecnologia para o gerenciamento. 2. ed. São Paulo: Saraiva, 2012.

COSTA, E. R. **Banco de dados relacionais**. 64f. Monografia (Graduação em Processamento de Dados), Faculdade de Tecnologia de São Paulo – FATEC, São Paulo, 2011. Disponível em: http://www.fatecsp.br/dti/tcc/tcc0025.pdf. Acesso em: 06 abr. 2020.

DATE, C. J. **Introdução a sistemas de banco de dados**. 8. ed. Rio de Janeiro: Campus, 2004.

ELMASRI, R.; NAVATHE S. B. **Sistema de banco de dados**. 7. ed. São Paulo: Pearson Addison Wesley, 2018. Disponível em: https://plataforma.bvirtual.com.br/Leitor/Publicacao/168492/pdf/0. Acesso em: 15 maio 2020.

GIL, A. C. **Como elaborar projetos de pesquisa**. 6. ed. São Paulo: Editora Atlas S.A., 2017.

HONG, Wei **The postgres and illustra codelines**. Nova York: ACM Books, 2018. Disponível em: https://dl.acm.org/doi/full/10.1145/3226595.3226623. Acesso em: 12 abr. 2020.

JOÃO, B. N. **Sistemas de informação**. São Paulo: Pearson, 2012. Disponível em: https://plataforma.bvirtual.com.br/Leitor/Publicacao/3056/pdf/0. Acesso em: 04 abr. 2020.

MILANI, ANDRÉ, **MySQL - Guia do programador**. São Paulo: Novatec, 2006. Disponível em: https://books.google.com.br/books?id=81EwMDA-pC0C&printsec=frontcover&hl=pt-BR#v=onepage&q&f=false. Acesso em: 27 out. 2020.

KENLER, E.; RAZZOLI, F. **MariaDB Essentials**. Birmingham: Packt Publishers, 2015. Disponível em: https://books.google.com.br/books?hl=pt-BR&lr=&id=VPh_CwAAQBAJ&oi=fnd&pg=PP1&dq=MariaDB&ots=v3V6nRo9-4&sig=ppMdmin6pglK6Zr0l7rCyjFlqWU#v=onepage&q=MariaDB&f=false. Acesso em: 12 abr. 2020.

KERR, E. S. **Gerenciamento de requisitos**. São Paulo: Editora Pearson, 2015.

MEDEIROS, E. **Desenvolvendo de software com UML 2.0.** São Paulo: Pearson Education, 2004.

MEDEIROS, L. F. **Banco de dados:** princípios e práticas. Curitiba: Editora Intersaberes, 2013. Disponível em: https://plataforma.bvirtual.com.br/Acervo/Publicacao/6289. Acesso em: 05 abr. 2020.

MELLO, R. B. **Migração entre sistemas gerenciadores de banco de dados**. 98f. Monografia (Graduação), Instituto Municipal de Ensino Superior de Assis, Assis, 2009. Disponível em: https://cepein.femanet.com.br/BDigital/arqTccs/0611270062.pdf. Acesso em: 21 abr. 2020.

MENDONÇA, M. H. R. **Metodologia de migração de dados em um contexto de migração de sistemas legados**. 151f. Dissertação (Mestrado em Ciência da Computação), Universidade Federal de Pelotas - UFPEL, 2009. Disponível em: https://repositorio.ufpe.br/handle/123456789/1934. Acesso em: 12 mar. 2020.

OLIVEIRA, C. S; MARCELINO M. A. Metodologias e estratégias de migração de dados. **Sinergia**, v. 13, n. 3, p. 183-191, set./dez. 2012. Disponível em: https://ojs.ifsp.edu.br/index.php/sinergia/issue/view/19/38. Acesso em: 21 abr. 2020.

ROESCH, S. M. A. **Projetos de estágio e de pesquisa em administração:** guia para estágios, trabalhos de conclusão, dissertações e estudos de caso. 3. ed. São Paulo: Atlas, 2013.

ROMAN, A. V.; NOTARI, D. L. **Implantação de um sistema arquivístico unificado**. Artigo (Graduação em Sistemas de Informação), Universidade de Caxias do Sul - UCS, Bento Gonçalves, 2018. Disponível em: https://repositorio.ucs.br/xmlui/bitstream/handle/11338/3929/TCC%20Anderson%20Vidart%20Roman.pdf?sequence=1&isAllowed=y. Acessado em: 20 abr. 2020.

SANTOS NETO, P. A. S.; RODRIGUES NETO, J.; RIBEIRO JÚNIOR, F. C.; OLIVEIRA, P. A. **Requisitos para ferramentas de migração de dados**. In: IX Simpósio Brasleiro de Sistemas de Informação – IX SBSI, 9, 2013, João Pessoa, PB.

Anais... Porto Alegre: Sociedade Brasileira de Computação, 2013. Disponível em: https://sol.sbc.org.br/index.php/sbsi/article/view/5749. Acesso em: 26 abr. 2020.

SETZER, V. W. Dado, Informação, Conhecimento e Competência. **Universidade de São Paulo**, 25 maio 2015. Disponível em: https://www.ime.usp.br/~vwsetzer/dado-info.html. Acessado em: 29 mar. 2020

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Sistema de banco de dados**. 3. ed. São Paulo: Makron Books, 1999.

THOMSEN, C.; PEDERSEN, T. B. **Pygrametl: a powerful programming framework for extract–transform–load programmers**. Hong Kong: ACM Books, 2009. Disponível em: https://dl.acm.org/doi/abs/10.1145/1651291.1651301. Acesso em: 21 abr. 2020.

VAZQUEZ, Carlos Eduardo e SIMÕES, Guilherme Siqueira. **Engenharia de requisitos software orientado ao negócio.** São Paulo: Fatto, 2016.

VELIMENETI, S. Data migration from legacy systems to modern database. **Culminating Projects in Mechanical and Manufacturing Engineering**, St. Cloud, n. 54, ago. 2016. Disponível em: https://repository.stcloudstate.edu/cgi/viewcontent.cgi?article=1054&context=mme_etds. Acesso em: 21 abr. 2020.

VOGEL, L. **Desenvolvimento de uma ferramenta para auxiliar na migração de dados entre sistemas ERP**. 82f. Monografia (Graduação em Sistemas de Informação), Universidade do Vale do Taquari – Univates, Lajeado, 2019. Disponível em: https://www.univates.br/bdu/bitstream/10737/2716/1/2019LeandroVogel.PDF. Acesso em 21 abr. 2020.